Пацей Н.В., Занько Д.В.

Конструирование программ и языки программирования

Практикум
В 2 частях
1 часть
для студентов специальности
1-40 01 02 (1-40 01 02-03)
«Информационные системы и технологии»

Минск 2005

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Пацей Н.В., Занько Д.В.

Конструирование программ и языки программирования

Практикум
В 2 частях
1 часть
для студентов специальности
1-40 01 02 (1-40 01 02-03)
«Информационные системы и технологии»

Минск БГТУ 2005
УДК 681.3.06
ББК
П

Рассмотрено и рекомендовано к изданию редакционно-издательским советом университета

Н.В. Пацей, Д.В. Занько

Рецензенты:
профессор кафедры мат. обеспечения АСУ БГУ, док. техн. наук *И. В. Совпель*;
доцент кафедры ИТАС БГУИР, канд. техн. наук *О. В. Герман*

Пацей Н.В., Занько Д.В. **Конструирование программ и языки программирования.** Практикум. Часть 1. для студентов специальности 1-40 01 02 (1-40 01 02-03) «Информационные системы и технологии». – Мн.: БГТУ, 2005. – 000 с.

ISBN

В пособии в систематизированном виде рассматриваются основные понятия и описываются возможности языка C/C++, позволяющие разрабатывать программы в соответствии с парадигмой процедурного программирования по дисциплине «Конструирование программ и языки программирования», приводятся задачи для выполнения.

УДК 681.3.08
ББК

ISBN

технологический университет», 2005

УДК 681.3.06
ББК
П

Рассмотрено и рекомендовано к изданию редакционно-издательским советом университета

Н.В. Пацей, Д.В. Занько

Рецензенты:
старший преподаватель кафедры полиграфии БГТУ *Н. Б. Каледина*;
доцент кафедры ИТАС БГУИР, канд. техн. наук *О. В. Герман*

Пацей Н.В., Занько Д.В. **Конструирование программ и языки программирования.** Практикум. Часть 1. для студентов специальности 1-40 01 02 (1-40 01 02-03) «Информационные системы и технологии». – Мн.: БГТУ, 2005. – 000 с.

ISBN

В пособии в систематизированном виде рассматриваются основные понятия и описываются возможности языка C/C++, позволяющие разрабатывать программы в соответствии с парадигмой процедурного программирования по дисциплине «Конструирование программ и языки программирования», приводятся задачи для выполнения.

УДК 681.3.08
ББК

ISBN

ВВЕДЕНИЕ

Конструирование программ – один из важнейших разделов современной информатики. Развитие программно-аппаратных средств, появление новых стилей и технологий программирования, не снижают, а повышают уровень требований к алгоритмической культуре. Методика конструирования программ легко переносится на другие процедурные языки программирования, в том числе и объектно-ориентированные.

Программирование задач рассматривается на процедурно-ориентированном языке программирования C/C++ . Язык программирования C/C++ – это универсальный язык, который позволяет разрабатывать программы в соответствии с разными парадигмами: процедурным программированием, объектно-ориентированным (ООП), параметрическим. C++ является расширением языка C: любая конструкция на языке C является корректной в C++; в то же время обратное неверно. Наиболее значительные нововведения, присутствующие в C++, касаются классов, объектов и ООП. Тем не менее, имеются и другие усовершенствования, связанные со способами организации ввода/вывода и написанием комментариев.

В данной работе рассматриваются основные возможности языка C/C++ и их применение при процедурном программировании по дисциплине «Конструирование программ и языки программирования» для специальности «Информационные системы и технологии».

Практикум содержит теоретический материал по темам: интегрированная среда разработки MS Visual C++, основные элементы языка, структура программы и типы данных, операции и операторы языка, массивы, функции, указатели и операции над ними, динамическое распределение памяти, файлы и их обработка, и варианты заданий для выполнения, необходимые для выполнения 17 работ.

Практикум №1. ЗНАКОМСТВО С C/C++. ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ MICROSOFT VISUAL C++

Структура программы

Все программы на C/C++ должны начинаться с функции, называемой main(). Она выглядит так:

```
void main() { }
```

main – это имя главной функции программы. С функции main всегда начинается выполнение. В круглых скобках перечисляются аргументы или параметры функции (в данном случае у функции main аргументов нет). У функции может быть результат или возвращаемое значение. Если функция не возвращает никакого значения, то это обозначается ключевым словом void. В фигурных скобках записывается тело функции – действия, которые она выполняет. Пустые фигурные скобки означают, что никаких действий не производится.

В общем случае программа на С имеет следующую структуру:

```
#директивы препроцессора
. . . . . .
#директивы препроцессора
функция а ( )
        операторы
функция в ( )
        операторы
void main ( )     //функция, с которой начинается выполнение программы
        операторы
                описания
                присваивания
                функция
                пустой оператор
                составной
                выбора
                циклов
                перехода
```

Разработка, компиляция и выполнение программы

Для начала разработки необходимо создать средствами Visual C++ файл рабочего пространства и проекта, либо загрузить уже имеющееся рабочее пространство и, если в нем есть несколько проектов, выбрать один из них.

- Загрузка проекта в рабочее пространство

В меню **File** выберите пункт **Open Workspace** для загрузки с диска информации о рабочем пространстве или **Recent Workspaces**

для загрузки одного из 4 последних использованных рабочих пространств. Каждое рабочее пространство содержит один или более проектов разрабатываемых программ. Если в рабочем пространстве один проект, то он будет выбран для редактирования, если их несколько, то – последний редактируемый проект для этого рабочего пространства.

- Создание нового проекта программы

Для создания нового проекта нужно выбрать команду **File/New**. Так как ничего в среде разработки нет, то система предложит вам создать новый проект и появится диалог, представленный на рис.1. В закладке *Projects* в списке различных типов выполняемых файлов выберите *Win32 Consol Application*. Убедитесь, что отмечена кнопка **Create new workspace**. Затем следует набрать имя проекта (например, *2*) в поле *Project name* и имя каталога, в котором будут храниться все файлы, относящиеся к данному проекту, в поле *Location*. После этого нажмите кнопку **OK**.

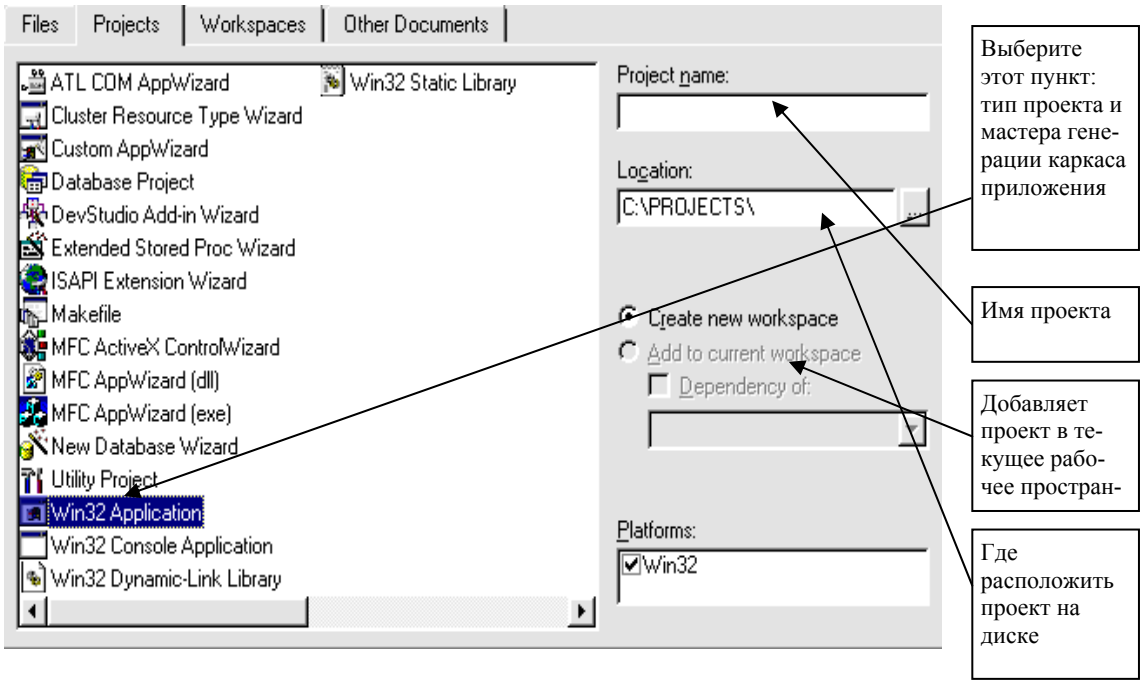


Рис. 1.

Теперь необходимо создать файл. Опять выберите команду **File/New**. В появившемся диалоге (рис.2) в закладке *File* отметьте *C++ Source File*. По умолчанию новый файл будет добавлен к текущему проекту (например, *2*), в чем можно убедиться, взглянув на поле

Add to project. В поле *File name* нужно ввести имя файла (например, *file1.cpp*). Расширение *.cpp* – стандарт для файлов с исходными текстами на языке C++. Поле *Location* должно показывать на каталог с рабочими проектами (например *C:\PROJECTS)*. Нажмите кнопку **OK**, на экране появится пустой файл. Наберите текст программы.

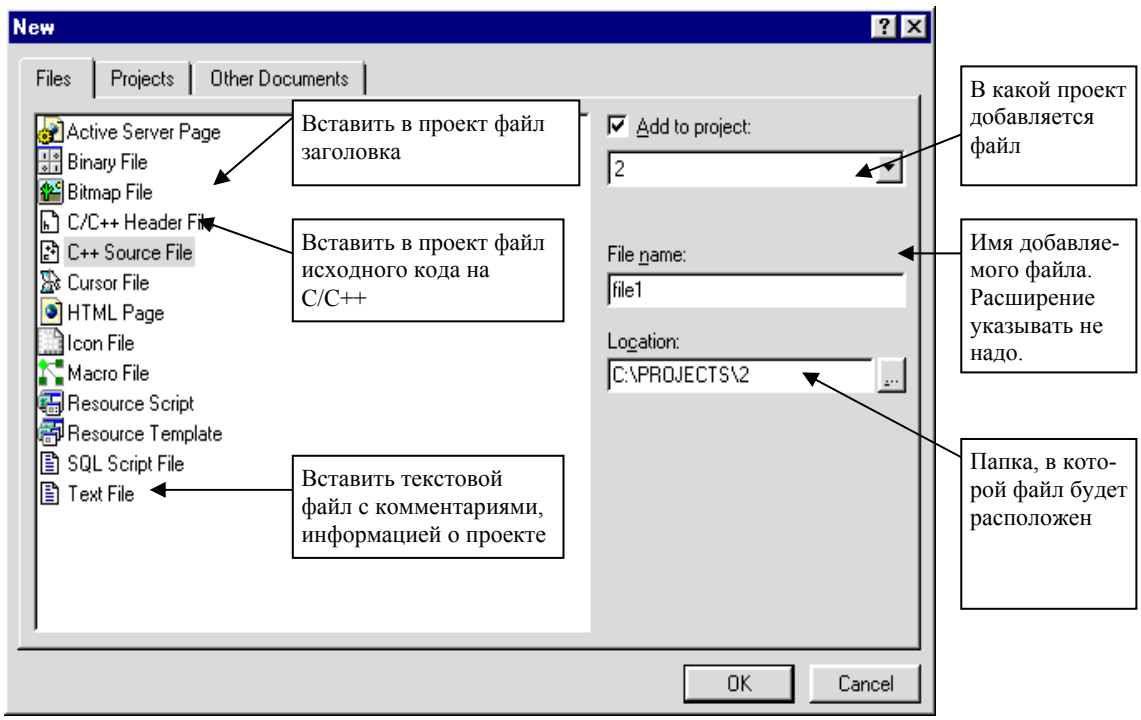


Рис. 2.

В процессе работы возможно понадобится добавить в проект файлы других исходных текстов, заголовков, ресурсов и т.д. Для этого в меню **Project** выберите пункт **Add to Project**, а из него выберите подпункт **New**.

Редактор среды Visual C++ многооконный, вы можете открыть сколько угодно окон с текстами файлов, заголовков и программ. Для переключения между этими окнами используется комбинация **Ctrl+F6**.

- Просмотрщик рабочего пространства Workspace Viewer

Workspace Viewer – просмотрщик классов, функций, глобальных переменных, файлов заголовков и исходных кодов, ресурсов которые используются в данном проекте, по умолчанию расположенный в левом углу окна среды разработки (рис.3).

Переключаясь от одной вкладки к другой, вы можете просмотреть список полей и методов внутри него: достаточно щелкнуть на плюсе перед именем класса и будет выведен список методов и переменных классов и также список классов, структур, перечислений определенных внутри класса.

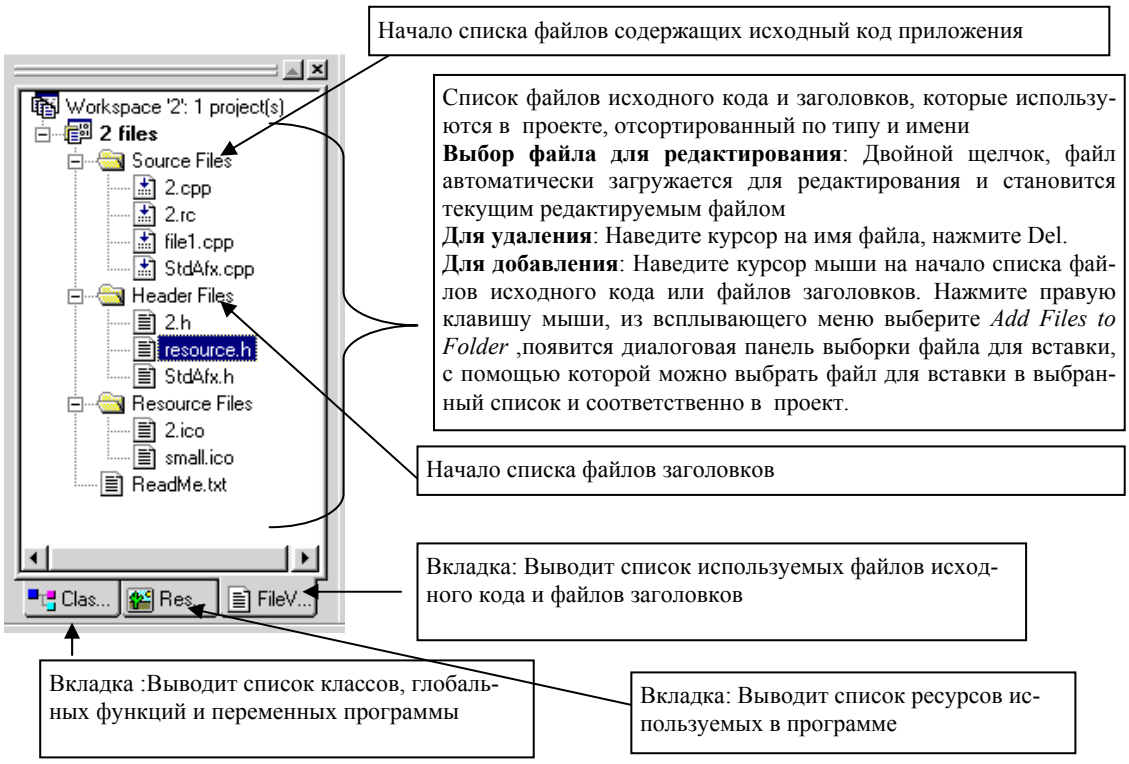


Рис. 3

- Компиляция, построение и выполнение программы

После набора текста программы ее необходимо откомпилировать и запустить. Программу можно компилировать по частям, то есть из каждого файл с исходными кодами программ (файлы с расширениями **.c* или **.cpp*) может быть создан объектный файл, который в дальнейшем может быть обработан компоновщиком. Таким образом, нет необходимости проводить длительную компиляцию всей программы, когда вы внесли изменение всего в один из файлов исходного текста программы.

Компиляция выполняется с помощью меню **Build**. Выберите пункт **Build имя файла** *(2.exe)* (этому пункту меню соответствует функциональная клавиша **F7**). В нижней части экрана появятся сооб-

щения компиляции. Если Вы сделали ошибку, двойной щелчок мышью по строке с ошибкой переведет курсор в окне текстового редактора на соответствующую строку кода. После исправления всех ошибок и повторной компиляции система выдаст сообщение об успешной компиляции и компоновке (вы увидите сообщение *Linking*). Аналогичные команды доступны через панель иснструментов рис.4.

Готовую программу можно выполнить с помощью меню **Build**, пункт **Execute имя файла** *(2.exe)*. То же самое можно сделать, нажав комбинацию **Ctrl+F5**. На экране монитора появится консольное окно, и в нем будут результаты работы программы. Затем появится надпись *"Press any key to continue"*. Эта надпись означает, что программа выполнена и лишь ожидает нажатия произвольной клавиши, чтобы закрыть консольное окно.

В режиме отладки вам доступны следующие команды (рис. 4): **Step Over -** переход к выполнению следующей команде без захода во внутрь функций (**F10)**; **Step Into -** переход к выполнению следующей команде с заходом во внутрь функций (**F11)**; **QuickWatch** - просмотр значений переменной или класса, на имя которых наведен курсор мыши **(Shift+F9)**.

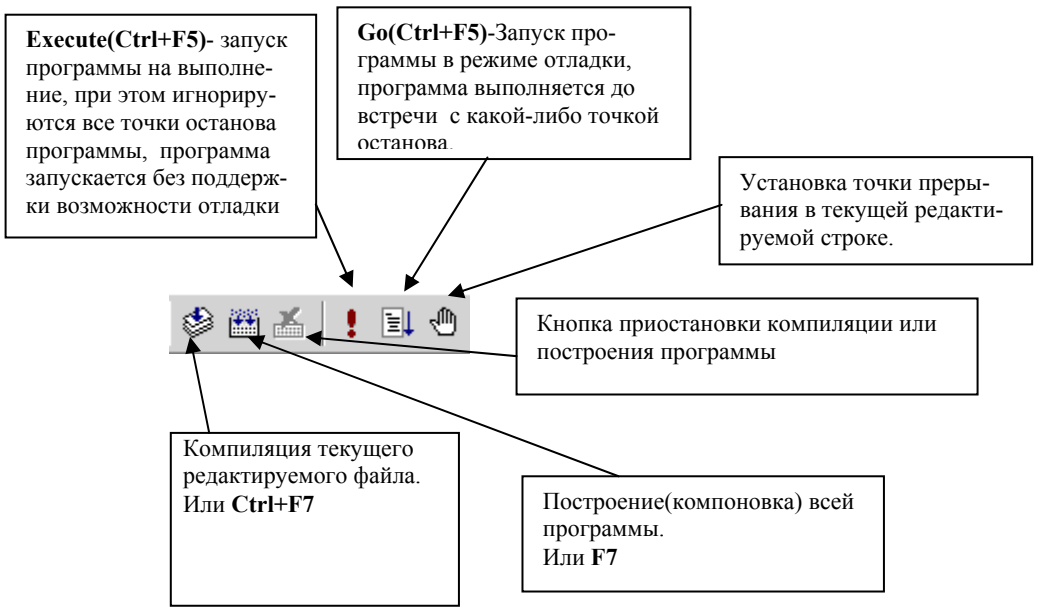


Рис. 4

Элементы языка C/C++

- Используемые символы

Множество символов можно разделить на пять групп.

1. Символы, используемые для образования ключевых слов и идентификаторов (табл. 1). Одинаковые прописные и строчные буквы считаются различными символами, так как имеют различные коды.

Таблица 1

Прописные буквы латинского алфавита	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Строчные буквы латинского алфавита	a b c d e f g h i j k l m n o p q r s t u v w x y z
Символ подчеркивания	_

2. Группа прописных и строчных букв русского алфавита и арабские цифры (табл. 2).

Таблица 2

Прописные буквы русского алфавита	А Б В Г Д Е Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ы Ь Э Ю Я
Строчные буквы русского алфавита	а б в г д е ж з и к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я
Арабские цифры	0 1 2 3 4 5 6 7 8 9

3. Знаки нумерации и специальные символы (табл. 3). Они используются для организации процесса вычислений и для передачи компилятору определенного набора инструкций.

Таблица 3

Символ	**Наименование**	**Символ**	**Наименование**
,	запятая	)	круглая скобка правая
.	точка	(	круглая скобка левая
;	точка с запятой	}	фигурная скобка правая
:	двоеточие	{	фигурная скобка левая
?	вопросительный знак	<	меньше
'	апостроф	>	больше
!	восклицательный знак	[	квадратная скобка
|	вертикальная черта	]	квадратная скобка
/	дробная черта	#	номер
\	обратная черта	%	процент
~	тильда	&	амперсанд
*	звездочка	^	логическое не
+	плюс	=	равно
-	мину	"	кавычки

4. Управляющие и разделительные символы: пробел, символы табуляции, перевода строки, возврата каретки, новая страница и новая строка. Эти символы отделяют друг от друга объекты, определяемые пользователем, к которым относятся константы и идентификаторы.

5. Кроме выделенных групп символов в языке C/C++ широко используются так называемые, управляющие последовательности, т.е. специальные символьные комбинации, используемые в функциях ввода и вывода информации. Управляющая последовательность строится на основе использования обратной дробной черты (\) (обязательный первый символ) и комбинации латинских букв и цифр (табл. 4).

Таблица 4

Управляющая последовательность	**Наименование**	**Шеснадцатеричная замена**
\a	Звонок	007
\b	Возврат на шаг	008
\t	Горизонтальная табуляция	009
\n	Переход на новую строку	00A
\v	Вертикальная табуляция	00B
\r	Возврат каретки	00C
\f	Перевод формата	00D
\"	Кавычки	022
\'	Апостроф	027
\0	Ноль-символ	000
\\	Обратная дробная черта	05C
\ddd	Символ набора кодов в восьмеричном представлении	
\xddd	Символ набора кодов в шестнадцатеричном представлении	

В строковых константах всегда обязательно задавать все три цифры в управляющей последовательности. Например, отдельную управляющую последовательность \n (переход на новую строку) можно представить как \010 или \xA, но в строковых константах необходимо задавать все три цифры, в противном случае символ или символы следующие за управляющей последовательностью будут рассматриваться как ее недостающая часть.

- Константы

Константами называются перечисление величин в программе. Разделяют четыре типа констант: целые константы, константы с плавающей запятой, символьные константы и строковые литералы.

Целая константа представляет целую величину в одной из следующих форм: десятичной, восьмеричной или шестнадцатеричной. Десятичная константа состоит из одной или нескольких десятичных цифр, причем первая цифра не должна быть нулем (в противном случае число будет воспринято как восьмеричное). Восьмеричная константа состоит из обязательного нуля и одной или нескольких восьмеричных цифр. Шестнадцатеричная константа начинается с обязательной последовательности 0x или 0X и содержит одну или несколько шестнадцатеричных цифр. Примеры целых констант:

```
Десятичная      Восьмеричная      Шестнадцатеричная
константа       константа         константа
    16              020               0x10
   127              0117              0x2B
   240              0360              0XF0
```

Если требуется сформировать отрицательную целую константу, то используют знак «–» перед записью константы (который будет называться унарным минусом): `-0x2A, -088, -16.`

Каждой целой константе присваивается тип, определяющий преобразования, которые должны быть выполнены, если константа используется в выражениях. Тип константы определяется следующим образом:

– десятичные константы рассматриваются как величины со знаком, и им присваивается тип int (целая) или long (длинная целая) в соответствии со значением константы.

– восьмеричным и шестнадцатеричным константам присваивается тип int, unsigned int (беззнаковая целая), long или unsigned long в зависимости от значения константы согласно табл 5.

Таблица 5

Диапазон шестнадцатеричных констант	**Диапазон восьмеричных констант**	**Тип**
0x0 - 0x7FFF	0 - 077777	int
0X8000 - 0XFFFF	0100000 - 0177777	unsigned int
0X10000 - 0X7FFFFFFF	0200000 - 017777777777	long
0X80000000 - 0XFFFFFFFF	020000000000 - 037777777777	unsigned long

Для того чтобы любую целую константу определить типом long, достаточно в конце константы поставить букву "l" или "L": `5l, 6l, 128L, 0105L, OX2A11L.`

Константа с плавающей точкой – десятичное число, представленное в виде действительной величины с десятичной точкой или

экспонентой. Формат имеет вид:

```
[ цифры ].[ цифры ] [ E|e [+|-] цифры ].
```

Число с плавающей точкой состоит из целой и дробные части и (или) экспоненты. Константы с плавающей точкой представляют положительные величины удвоенной точности (имеют тип double). Для определения отрицательной величины необходимо сформировать константное выражение, состоящее из знака минуса и положительной константы: `115.75, 1.5E-2, -0.025, .075, -0.85E2.`

Символьная константа – представляется символом заключенном в апострофы: `' ', 'Q', '\n', '\\'`. Значением символьной константы является числовой код символа.

Строковая константа (литерал) – последовательность символов (включая строковые и прописные буквы русского и латинского а также цифры) заключенные в кавычки ("): `"город Тамбов", "YZPT КОД"`.

Символы строковой константы сохраняются в области оперативной памяти. В конец каждого строкового литерала компилятором добавляется нулевой символ, представляемый управляющей последовательностью \0.

Описание констант начинается с ключевого слова const, далее указывается тип и значение:

```
const int Size=2;
```

Поскольку константе ничего нельзя присвоить, она должна быть инициализирована. Описание чего-нибудь как const гарантирует, что его значение не изменится в области видимости:

```
Size=2;      //ошибка
```

- Идентификаторы

Идентификатором называется последовательность цифр и букв, а также специальных символов, при условии, что первой стоит буква или специальный символ. В качестве специального символа может использоваться символ подчеркивание (_). Два идентификатора для образования которых используются совпадающие строчные и прописные буквы, считаются различными: `abc, ABC, A128B, a128b`.

Компилятор допускает любое количество символов в идентификаторе, хотя значимыми являются первые 31 символ. Идентификатор создается на этапе объявления переменной, функции, структуры и т.п. после этого его можно использовать в последующих операторах разрабатываемой программы. Идентификатор не должен совпадать с

ключевыми словами, с зарезервированными словами и именами функций библиотеки компилятора языка C/C++.

- Ключевые слова

Ключевые слова – это зарезервированные идентификаторы, которые наделены определенным смыслом. Их можно использовать только в соответствии со значением известным компилятору языка.

Список ключевых слов:

```
auto      double    int    struct  break   else    long    switch
register  typedef   char   extern  return  void    case    float
unsigned  default   for    signed  union   do      if      sizeof
volatile  continue  enum   short   while   _asm    fortran near
far       cdecl     huge   paskal  interrupt
```

- Комментарии

Комментарий – это набор символов, которые игнорируются компилятором. Введение комментария начинается с символов /* и заканчивается символами */. Все, что помещено между ними игнорируется:

```
/*Эта программа выводит сообщение на экран*/
```

Комментарии /* */ не могут быть вложенными. В C++ используется пара символов //, указывающая начало строки комментария. В этом случае концом комментария считается конец строки, так что нет необходимости отмечать его специальным символом:

```
//Эта программа выводит сообщение на экран
```

Этот способ наиболее полезен для коротких комментариев.

Контрольные вопросы

1. Какое самое длинное локальное имя можно использовать в программе?

2. Есть ли какие-нибудь ограничения на символы, которые можно употреблять в имени?

3. Определить тип констант: `134L; 123.5L; 123.5`.

4. Что означает квалификатор `const char msg[]="предупреждение"`?

5. Можно ли задать идентификатор `near`?

Практикум №2. ТИПЫ ДАННЫХ И ПЕРЕМЕННЫЕ. ОПЕРАТОРЫ ВВОДА-ВЫВОДА

Краткие теоретические сведения

- Основные типы

Программа оперирует информацией, представленной в виде различных объектов и величин. С точки зрения архитектуры компьютера, переменная – это символическое обозначение ячейки оперативной памяти программы, в которой хранятся данные. Доступ к значению возможен через имя переменной, а доступ к участку памяти – по его адресу. Каждая переменная перед использованием в программе должна быть объявлена, т. е. ей должна быть выделена память. Размер участка памяти, выделяемой для переменной и интерпретация содержимого зависят от типа, указанного в определении переменной. Тип переменной изменить нельзя. Простейшая форма объявления переменных:

```
тип список_имен_переменных;
```

Переменным можно присваивать начальные значения, явно указывая их в определениях, этот прием называется инициализацией:

```
тип имя_переменной = начальное_значение;
```

Примеры:

```
float pi = 3.14 , cc=1.3456;
unsigned int year = 1999;
```

Целый тип – это обобщенный тип, представление которого зависит от операционной системы и типа процессора (табл. 6).

Таблица 6

Тип данных	Название	Размер, бит	Диапазон значений
unsigned int	беззнаковый целый	16	0 . . 65535
short int (short)	короткий целый	16	-32768 . . 32767
unsigned short	беззнаковый короткий целый	16	0 . . 65535
int	целый	16	-32768 . . 32767
unsigned long	беззнаковый длинный целый	32	0 . . 4294967295
long	длинный целый	32	-214748348 . . 2147483647

Пример:

```
int      Stavka;
int      Symma;
Stavka = 300;
Symma = Stavka * 2;
```

Для хранения чисел с дробной частью используется вещественный тип (табл. 7).

Таблица 7

Тип данных	Название	Размер, бит	Диапазон значений
float	вещественный одинарной точности	32	3.4E-38 . . 3.4E+38
double	вещественный двойной точности	64	1.7E-308 . . 1.7E+308
long double	вещественный максимальной точности	80	3.4E-4932 . . 1.1E+4932

Символьный тип – char предназначен для хранения одного символа, поэтому его размер – один байт (табл. 8).

Таблица 8

Тип данных	Название	Размер, бит	Диапазон значений
unsigned char	беззнаковый целый длиной не менее 8 бит	8	0 . . 255
char	целый длиной не менее 8 бит	8	-128 . . 127

Пример:

```
char         A;
char         B;
A = 'D';
B = '!';
```

Для представления строки символов в С используют массивы типа char:

```
char         B[11];
char         C[24];
```

Так же можно использовать указатель на символы:

```
char *msg;
msg = "Привет, студент";
```

Именованная совокупность однородных данных называется массивом и представляет собой производный тип данных. Каждый элемент массива хранится в отдельной области памяти и имеет собственный номер или индекс (начиная с нуля):

```
int B[3];
B[0] = 10; B[1] = 20; B[2] = 30;
```

Тип void (пустой) синтаксически ведет себя как основной тип. Однако использовать его можно только как часть производного типа, объектов типа void не существует.

По области видимости переменные можно разделить на три группы. Переменная, определенная во всех модулях (файлах) программы – определяется при помощи ключевого слова **extern** и будет видна во всех точках программы (глобальная для всей программы). Переменная, определенная в одном из модулей (файле) перед текстами всех функций. Такая переменная будет глобальной для данного модуля, т.е. может быть использована во всех точках данного модуля. Переменная, определенная в данной функции может быть использована только в пределах данной функции (локальная).

По времени жизни переменные делятся на две группы: живущие в течении работы программы; переменные уничтожающиеся после выхода из функции.

Основные типы можно сочетать в присваиваниях и выражениях, при этом происходит неявное преобразование типов. Значения преобразуются так, чтобы информация не терялась. Присваивание значения одного типа переменной другого типа, представление которого содержит меньшее число бит, приводит к потере информации.

Явное преобразование типа дает значение одного типа для данного значения другого типа. Есть два способа записи явного преобразования типа. Каноническая запись приведения к типу: `(имя_типа) операнд` и функциональная запись: `имя_типа (операнд)`. Функциональная запись не может применяться для типов, которые не имеют простого имени. Например, чтобы преобразовать значение к указательному типу надо или использовать запись приведения:

```
char* p = (char*)0777;
```

или определить новое имя типа:

```
typedef char* Pchar;
char* p = Pchar(0777);
```

- Спецификаторы класса памяти

К спецификаторам класса памяти относятся:

```
auto
static
extern
register
```

Описания `auto`, `static` и `register` служат также в качестве определений в том смысле, что они вызывают резервирование нужного количества памяти. В случае `extern` должно присутствовать внешнее определение указываемых идентификаторов где-то вне функции, в которой они описаны. Описание `register` используется для часто используемых описаний. Если описание не содержит спецификатора класса памяти, то считается, что он имеет значение `auto`, если описание находится внутри некоторой функции, и `extern` в противном случае. Статические `static` переменные могут быть либо внутренними, либо внешними. Внутренние статические переменные, как и автоматические, являются локальными для некоторой функции, но, в отличие от автоматических, они остаются существовать, а не появляются и исчезают вместе с обращением к этой функции. Это означает, что внутренние статические переменные обеспечивают постоянное, недоступное извне хранение внутри функции. Внешние статические переменные определены в остальной части того исходного файла, в котором они описаны, они позволяют скрывать имена. Описание может содержать не более одного спецификатора класса памяти.

- Потоковый ввод и вывод

Обмен данными между программой и внешними устройствами осуществляется с помощью операций ввода-вывода. Классы ввода-вывода определены в файле заголовков `#include <iostream.h>`. Библиотека потоков ввода-вывода определяет три глобальных объекта: `cout`, `cin` и `cerr`. `cout` называется стандартным выводом, `cin` – стандартным вводом, `cerr` – стандартным потоком сообщений об ошибках.

Вывод осуществляется с помощью операции **>>**, ввод с помощью операции **<<**. Выражение

```
cout << "Пример вывода: " << 34;
```

напечатает строку *"Пример вывода"*, за которым будет выведено число *34*. Используя один стандартный поток вывода `cout`, можно отобразить несколько аргументов. Между собой аргументы разделяются операторами вставки:

```
int age;
age = 43;
cout << "Вам исполнилось " << age << " года.";
```

Выражение

```
int x;
cin >> x;
```

введет целое число с консоли в переменную *x* (для того, чтобы ввод произошел, нужно напечатать число и нажать клавишу **Enter**.)

Часто бывает необходимо вывести строку или число в определенном формате. Для этого используются манипуляторы – объекты особых типов, которые управляют тем, как обрабатываются последующие аргументы:

```
endl            – при выводе перейти на новую строку;
ends            – вывести нулевой байт (признак конца строки символов);
dec             – выводить числа в десятичной системе (по умолчанию);
oct             – выводить числа в восьмеричной системе;
hex             – выводить числа в шестнадцатеричной системе счисления;
setw(int n)     – установить ширину поля вывода в n символов (n целое);
setfill(int n) – установить символ-заполнитель; этим символом выводимое значение будет дополняться до необходимой ширины;
setprecision(int n) – установить количество цифр после запятой при выводе вещественных чисел;
setbase(int n)  - установить систему счисления для вывода чисел; n может принимать значения 0, 2, 8, 10, 16, причем 0 означает систему счисления по умолчанию, т.е. 10.
```

Для использования манипуляторов, их надо вывести в выходной поток:

```
int x = 53;
cout << "Десятичный вид:" << dec << x << endl
     << "Восьмеричный вид:" << oct << x << endl
     << "Шестнадцатеричный вид:" << hex << x << endl;
```

Аналогично используются манипуляторы с параметрами:

```
double x;
// вывести число в поле общей шириной 6 символов (3 цифры до запятой,
// десятичная точка и 2 цифры после запятой)
cout << setw(6) << setprecision(2) << x << endl;
```

Те же манипуляторы (за исключением `endl` и `ends`) могут использоваться и при вводе:

```
int x;
// ввести шестнадцатеричное число
cin >> hex >> x;
```

- Форматированный ввод и вывод

Обмен данными реализуется с помощью библиотеки функций ввода-вывода: `#include <stdio.h>`.

Функция `printf()` позволяет выводить на дисплей данные всех типов, работать со списком из нескольких аргументов и определять способ форматирования данных:

```
printf ( <форматная строка>,<список аргументов>);
```

<форматная строка> – строка символов, заключенных в кавычки, которая показывает, как должны быть напечатаны аргументы. Например:

```
printf ( "Значение числа Пи равно  %f\n", pi);
```

Форматная строка может содержать: символы печатаемые текстуально; спецификации преобразования; управляющие символы.

Каждому аргументу соответствует своя спецификация преобразования, которая начинается с символа процента (%), после которого стоит буква, указывающая тип данных:

```
%d -  десятичное целое число;
%f -  вещественное число типа float или double;
%c -  символ;
%s -  строка;
%u -  беззнаковое целое число;
%o -  целые числа в восьмеричной системе счисления;
%x -  целые числа в шестнадцатеричной системе счисления;
%e -  вещественное число в экспоненциальной форме.
```

Можно управлять перемещением курсора на экране и выполнять некоторые другие функции, используя управляющие коды, называемые escape-последовательностями. Последовательность начинается с символа обратной наклонной черты (\):

```
\n - перемещает курсор в начальную позицию следующей строки;
\t – перемещает курсор в следующую позицию табуляции экрана;
\r – выполняет «возврат каретки», перемещая курсор к началу той же
строки без перехода на следующую;
\b – передвигает курсор только на одну позицию влево.
```

Функция `scanf()` является многоцелевой функцией, дающей возможность вводить в компьютер данные любых типов. Указатели формата аналогичны тем, которые используются функцией `printf()`.

```
scanf ( <форматная строка>,<список аргументов>);
```

В качестве аргументов используются указатели. Например:

```
scanf(" %d%f ", &x,&y);
```

- Строковы и символьный ввод и вывод

Реализуется функцией `puts()`, которая осуществляет вывод информации на экран. Параметром может быть строка:

```
puts("Всем привет!");
```

строковая константа:

```
#define MESSAGE "Всем привет"
puts(MESSAGE);
```

или строковая переменная:

```
char greeting[]="Всем привет";
puts(geering);
```

Функция `putchar()` предназначена для вывода единичного символа на экран. Параметром функции может быть символьный литерал:

```
putchar('H');
```

символьная константа:

```
#define INITIAL 'H'
putchar(INITIAL);
```

или символьная переменная:

```
char letter;
letter='G';
putchar(letter);
```

Функция `gets()` вводит строку в переменную:

```
char name [60];
printf("Как вас зовут: ");
gets (name);
printf ("Привет, %s\n", name);
```

Функция `getchar()` вводит с клавиатуры единичный символ:

```
int letter;
letter = getchar();
```

Задания для выполнения

Таблица 9

1	Напишите программу ввода и вывода на экран вашего любимого стихотворения (латинский). Для построчного вывода используйте подходящую управляющую последовательность.
2	Стихотворный текст (в строке не более 80 символов) имеет четырехстрочную строфу. Записать его «лесенкой» (по одному слову в строке), вставляя пустую строку после каждого четверостишья.
3	Напишите программу вывода текса (символов) «елочкой»
4	Напишите программу вывода круга по центру консольного окна, закрашенного введенным сиволом.

Окончание таблицы 9

5	Напишите программу вывода треугольника по центру консольного окна, закрашенного введенным сиволом.
6	Напишите программу вывода звезды, закрашенную введенным сиволом.
7	Напишите программу вывода трапеции по центру консольного окна, закрашенную введенными сиволами.
8	Напишите программу вывода человечка на консольное окно.
9	Напишите программу вывода разнонаправленных стрелок, состоящих из введенного сивола.
10	Напишите программу диалога между пользователем и банкоматом, эмулирующей съем денег. Использовать операторы форматированного ввода-вывода.
11	Напишите программу диалога между пользователем и компьютером, эмулирующей резервирование авиабилета. Использовать операторы потокого ввода-вывода с манипуляторами.
12	Напишите программу диалога между пользователем и интернет-магазином, эмулирующей покупку компьютера. Использовать операторы потокого ввода-вывода с манипуляторами.
13	Напишите программу диалога между пользователем и компьютером, эмулирующей внесение записи в базу данных сотрудников предприятия. Использовать известные операторы ввода-вывода.
14	Напишите программу диалога между пользователем и компьютером, эмулирующей выбор диска в видеопрокате.
15	Напишите программу диалога «Знакомство».

Контрольные вопросы

1. Что такое объявления перменных?
2. Можно ли в объявлении инициализировать перменные?
3. Перечислите базовые типы данных?
4. Существуют ли объекты типа void?
5. Какую область действия имеют внешние пременные?
6. Приведите пример неявного преобразования типа данных?
7. Как в С реализуются операции ввода-ввывода? Перечислите известные вам операторы?
8. Что делат оператор: `cin>>n;`?
9. Какое необходимо ввести занчение x согласно следующей спецификации: `scanf("%s", &x);`?
10. В чем разница между функциями `puts()` и `putchar()`?

Практикум №3. ИСПОЛЬЗОВАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ И ВЫРАЖЕНИЙ ЯЗЫКА С/С++. ЛИНЕЙНЫЕ АЛОГРИТМЫ

Краткие теоретические сведения

- Основные операции

Основные унарные опреации (с одним операндом) приведены в табл. 10. Унарные операции выполняются справа налево.

Таблица 10

Операции	Назначение
&	получение адреса операнда
*	обращение по адресу (разыменование)
-	унарный минус, меняет знак арифметического операнда
~	поразрядное инвертирование внутреннего двоичного кода (побитовое отрицание)
!	логическое отрицание (НЕ). В качестве логических значений используется 0 - ложь и не 0 - истина, отрицанием 0 будет 1, отрицанием любого ненулевого числа будет 0.
++	увеличение на единицу: префиксная операция - увеличивает операнд до его использования, постфиксная операция увеличивает операнд после его использования.
--	уменьшение на единицу: префиксная операция - уменьшает операнд до его использования, постфиксная операция уменьшает операнд после его использования.
sizeof	вычисление размера (в байтах) для объекта того типа, который имеет операнд

Основные бинарные опреации (с двумя операндами) приведены в табл. 11. Бинарные опреации выполняются слева направо.

Таблица 11

Операции	Назначение	
+	бинарный плюс (сложение арифметических операндов)	Аддитивные
-	бинарный минус (вычитание арифметических операндов)	
*	умножение операндов арифметического типа	Мультипликативные
/	деление операндов арифметического типа (если операнды целочисленные, то выполняется целочисленное деление)	
%	получение остатка от деления целочисленных операндов	

<<	сдвиг влево битового представления значения левого целочисленного операнда на количество разрядов, равное значению правого операнда	Операции сдвига (определены только для целочисленных операндов)
>>	сдвиг вправо битового представления значения правого целочисленного операнда на количество разрядов, равное значению правого операнда	
&	поразрядная конъюнкция (И) битовых представлений значений целочисленных операндов	Поразрядные операции
|	поразрядная дизъюнкция (ИЛИ) битовых представлений значений целочисленных операндов	
^	поразрядное исключающее ИЛИ битовых представлений значений целочисленных операндов	
<	меньше, чем	Операции сравнения
>	больше, чем	
<=	меньше или равно	
>=	больше или равно	
==	равно	
!=	не равно	
&&	конъюнкция (И) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(1)	Логические бинарные операции
||	дизъюнкция (ИЛИ) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(1)	
=	присваивание, присвоить значение правого операнда левому	Операции присваивания и составного присваивания
+=	выполнить соответствующую операцию с левым операндом и правым операндом и присвоить результат левому операнду	
-=		
*=		
/=		
%=		
|=		
&=		
^=		
<<=		
>>=		

В отличие от унарных и бинарных операций в условных операциях (с тремя операндами) используется три операнда:

```
Выражение1 ? Выражение2 : Выражение3;
```

Первым вычисляется значение выражения 1. Если оно истинно, то вычисляется значение выражения 2, которое становится результатом. Если при вычислении выражения 1 получится 0, то в качестве результата берется значение выражения 3.

```
x<0 ? -x : x ; //вычисляется абсолютное значение x.
```

Приоритеты операций приведены в табл. 12.

Таблица 12

Ранг	Операции
1	( ) [ ] -> .
2	! ~ - ++ -- & * (тип) sizeof тип( )
3	* / % (мультипликативные бинарные)
4	+ - (аддитивные бинарные)
5	<< >> (поразрядного сдвига)
6	< > <= >= (отношения)
7	== != (отношения)
8	& (поразрядная конъюнкция «И»)
9	^ (поразрядное исключающее «ИЛИ»)
10	| (поразрядная дизъюнкция «ИЛИ»)
11	&& (конъюнкция «И»)
12	|| (дизъюнкция «ИЛИ»)
13	?: (условная операция)
14	= *= /= %= -= &= ^= |= <<= >>= (операция присваивания)
15	, (операция запятая)

- Выражения

Комбинация знаков операций и операндов, результатом которой является определенное значение, называется выражением. Каждый операнд в выражении может быть выражением. Значение выражения зависит от расположения знаков операций и круглых скобок в выражении, а также от приоритета выполнения операций. Каждое выражение состоит из одного или нескольких операндов, символов операций и ограничителей:

```
X * 12 + Y
val < 3
-9
```

Выражение, после которого стоит точка с запятой – это оператор-выражение.

- Операторы

Запись действий, которые должен выполнить компьютер, состоит из операторов. При выполнении программы операторы выполняются один за другим, за исключением операторов управления, которые могут изменить последовательное выполнение программы. Различают операторы объявления переменных, операторы управления и операторы-выражения.

Для организации множественных выражений, расположенных внутри круглых скобок используется оператор запятая. Выражение внутри скобок вычисляется слева направо. Например:

```
char  X,Y;
(X = Y, Y = getch()) //присваивает переменной X значение Y,
// считывает символ, вводимый с клавиатуры, и запоминает  его  в  Y
```

Основной источник операторов в программе – выражения. Любое из них, ограниченное символом ";", превращается в оператор. Простейшей формой оператора является пустой оператор:

```
;
```

Он не делает ничего. Однако он может быть полезен в тех случаях, когда синтаксис требует наличие оператора, а он не нужен.

Любая последовательность операторов, заключенная в фигурные скобки ({}), может выступать в любой синтаксической конструкции как один составной оператор (блок):

```
 {
a=b+2;
b++;
 }
```

Он позволяет рассматривать несколько операторов как один. Область видимости имени, описанного в блоке, простирается до конца блока.

- Макроподстановка

Макроподстановка определяется как:

```
#define      идентификатор  строка_текста
```

Директива обеспечивает замену встречающегося в тексте программы идентификатора на соответствующую строку текста, в том числе и с параметрами. Наиболее часто она применяется для символического обозначения константы, которая встречается многократно в различных частях программы. Например:

```
#define     SIZE  100
#define     FOR(i,n) for(i=0; i<n; i++)
```

Задания для выполнения

Таблица 13

Написать программу, вычисляющую следующие выражения, обеспечив при этом варианты: ввода данных с клавиатуры, инициализации данных в тексте программы, вывода данных на экран	
1	$y = \dfrac{1/\sqrt{x} + \cos e^x + \cos x^2}{\sqrt[3]{\ln(1+x)^2 + \sqrt{e^{\cos x} + \sin^2 \pi x}}}$
2	$y = (\cos e^x + \ln(1+x)^2 + \sqrt{e^{\cos x} + \sin^2 \pi x} + \sqrt{1/x} + \cos x^2)^{\sin x}$
3	$y = (\sin \pi x^2 + \ln x^2)^{(\sin x + e^{\cos x} + x^2)}$
4	$y = (x^2 + e^{\cos x} + \sin x)^{\sqrt{\sin \pi x^2 + \ln x^2}}$
5	$y = (\ln(1+x)^2 + \cos \pi x^3)^{\sin x} + (e^{x^2} + \cos e^x + \sqrt{1/x})^{1/x}$
6	$y = \dfrac{\sqrt[4]{\cos e^x + e^{x^2} + \sqrt{1/x}}}{(\cos \pi x^3 + \ln(1+x)^2)^{\sin x}}$
7	$y = \sin(\sin x + e^{\cos x} + x^2)\sqrt[4]{\sin \pi x^2 + \ln x^2}$
8	$y = \dfrac{\cos e^x + \sqrt{1/x} + \cos \pi x^3 + \ln(1+x)^2 + e^{x^2}}{\sqrt{\ln(1+x)^2 + \cos \pi x^3}}$
Написать программу решения задачи	
9	На биржевых торгах за 1 фунт стерлингов давали \$1.487, за франк — \$0.172 за немецкую марку — \$0.584, а за японскую йену — \$0.00955. Напишите программу, которая запрашивает денежную сумму в долларах, а затем вы водит эквивалентные суммы в других валютах.
10	Длина отрезка задана в дюймах (1 дюйм = 2,54 см). Перевести значение длины в метрическую систему, то есть выразить ее в метрах, сантиметрах и миллиметрах. Пример: 21 дюйм = 0 м 53 см 3,4 мм.
11	Заданы моменты начала и конца некоторого промежутка времени в часах, минутах и секундах (в пределах одних суток). Найти продолжительность этого промежутка в тех же единицах измерения.
12	Текущее время (часы, минуты, секунды) задано тремя переменными: *h*, *m*, *s*. Округлить его до целых значений минут и часов. Пример: 14 ч 21мин 45с преобразуется в 14 ч 22 мин или 14ч.
13	В такси одновременно сели три пассажира. Когда вышел первый пассажир, на счетчике было *p1* рублей; когда вышел второй — *p2* рублей. Сколько должен был заплатить каждый пассажир, если по окончании поездки счетчик показал *p3* рублей? Плата за посадку составляет *p0* рублей. Замечание: общая сумма оплаты пассажирами должна совпадать с показанием счетчика по окончании поездки. Рассмотрим крайние ситуации: все три пассажира вышли одновременно, первый и второй пассажиры «передумали ехать».

Продолжение таблицы 13

14	Коммерсант, имея стартовый капитал k рублей, занялся торговлей, которая ежемесячно увеличивает капитал на p%. Через сколько лет он накопит сумму s, достаточную для покупки собственного магазина?
15	Русские неметрические единицы длины: 1 верста = 500 саженей; 1 сажень - 3 аршина; 1 аршин = 16 вершков; 1 вершок - 44,45 мм. Длина некоторого отрезка составляет р метров. Перевести ее в русскую неметрическую систему.
16	За первый год производительность труда на предприятии возросла на p_1 %, за второй и третий – соответственно на p_2 и p_3 %. Найти среднегодовой прирост производительности (в процентах). Замечания: если ежегодный прирост постоянен, то и среднегодовой прирост р такой же: $p = p_1 = p_2 = p_3$ Общий прирост за 3 года в общем случае составит $p_\Sigma = \left[\left(1+\frac{p_1}{100\%}\right)\left(1+\frac{p_2}{100\%}\right)\left(1+\frac{p_3}{100\%}\right)-1\right]\cdot 100\%$. Тот же результат можно получить при среднегодовом приросте $p_\Sigma = \left[\left(1+\frac{p}{100\%}\right)-1\right]\cdot 100\%$ Остается найти величину p.
17	Целой переменной s присвоить сумму цифр трехзначного целого числа k.
18	Определить число, полученное выписыванием в обратном порядке цифр заданного четырехзначного числа.
19	Определить, есть ли среди цифр заданного трехзначного числа, одинаковые.
20	Дано действительное число a. Не пользуясь никакими операциями кроме умножения, получить a^3 и a^{10} за 4 операции.
21	Найти корни квадратного уравнения, заданного своими коэффициентами, с положительным дискриминантом; подстановкой в уравнение убедиться в погрешности вычислений.
22	Найти координаты вершины параболы $y=ax^2+bx+c$.
23	Найти объем пирамиды, построенной на векторах A, B, C, как на сторонах.
24	Треугольник ABC задан длинами своих сторон. Найти длину высоты, опущенной из вершины A. Расмотреть крайние случаи: сумма двух сторон равна третьей; одна из сторон равна нулю.
25	Треугольник задается координатами своих вершин на плоскости: $A(x_1,y_1),B(x_2,y_2),C(x_3,y_3)$ Найти площадь треугольника ABC.
26	Треугольник задается координатами своих вершин на плоскости: $A(x_1,y_1),B(x_2,y_2),C(x_3,y_3)$. Найти длину и основание высоты, опущенной из вершины A на сторону BC.
27	Треугольник задается координатами своих вершин на плоскости: $A(x_1,y_1),B(x_2,y_2),C(x_3,y_3)$. Найти сумму длин медиан треугольника ABC.

Окончание таблицы 13

28	Заданы уравнения двух пересекающихся прямых на плоскости: $y=k_1x+b_1$, $y=k_2x+b_2$. Найти (в градусах и минутах) угол между ними, используя формулу: $tg\varphi = \frac{(k_2 - k_1)}{(1 + k_1 k_2)}$.
29	У квадрата ABCD на плоскости известны координаты двух противоположных вершин – точек A и C. Найти координаты точек B и D. Замечание: расположение квадрата произвольно; его стороны не обязательно параллельны координатным осям.
30	Функция $y = sin(x)$ на отрезке $[0,\pi/2]$ аппроксимируется разложением: $y=x-x^3/6+x^5/120$. Для заданного значения аргумента x вычислить y по этой формуле и сравнить с точным значением, вычисленным с помощью стандартной функции *sin*.

Контрольные вопросы

1. Чему равно выражение `sizeof("asdf")`?
2. Что такое выражения и где их можно использовать?
3. Зачем при записи операций используют круглые скобки?
4. В чем разница между `--count` и `count--`?
5. Истинно ли следующее утверждение: в операции присваивания величина, стоящая слева от знака равенства, всегда равна величине, стоящей справа от знака равенства?

Практикум №4. РАЗЫВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

Краткие теоретические сведения

Все операторы управления могут быть условно разделены на следующие категории:

- условные операторы, к которым относятся оператор условия if и оператор выбора switch;
- операторы цикла (for,while,do while);
- операторы перехода (break, continue, return, goto).

- Условный оператор if

Формат оператора:

```
if <выражение-условие> оператор-1; [else оператор-2;] //полная форма
if <выражение-условие> оператор-1;                    //сокращенная форма
```

Выполнение оператора `if` начинается с вычисления <выражения-условия>. В качестве <выражения-условия> могут использоваться арифметическое выражение, отношение и логическое выражение. Далее выполнение осуществляется по следующей схеме: если выражение истинно (т.е. отлично от 0), то выполняется оператор-1, если выражение ложно (т.е. равно 0), то выполняется оператор-2, если выражение ложно и отсутствует оператор-2 (в квадратные скобки заключена необязательная конструкция), то выполняется следующий за `if` оператор. Пример:

```
if (i < j)   i++;
else {  j = i-3;   i++;  }
```

После выполнения оператора `if` значение передается на следующий оператор программы, если последовательность выполнения операторов не будет принудительно нарушена.

Допускается использование вложенных операторов `if`. Оператор `if` может быть включен в конструкцию `if` или в конструкцию `else` другого оператора `if`. Рекомендуется группировать операторы и конструкции, используя фигурные скобки. Если же фигурные скобки опущены, то компилятор связывает каждое ключевое слово `else` с наиболее близким `if`, для которого нет `else` :

```
int t=2, b=7, r=3;
   if (t>b)
      {
       if (b < r)  r=b;
      }
   else r=t; // r станет равным 2
return (0);
```

Если в программе опустить фигурные скобки, то получится:

```
int t=2,b=7,r=3;
  if ( a>b )
     if ( b < c ) t=b;
     else       r=t; //r получит значение равное 3
return (0);
```

Следующий фрагмент иллюстрирует вложенные операторы `if`:

```
char ZNAC;
int x,y,z;
   :
      if (ZNAC == '-') x = y - z;
      else  if (ZNAC == '+') x = y + z;
            else if (ZNAC == '*') x = y * z;
                  else if (ZNAC == '/') x = y / z;
                        else ...
```

Конструкции использующие вложенные операторы `if`, являются громоздкими и не всегда достаточно надежными.

- Оператор выбора switch

Другим способом организации выбора из множества различных вариантов является использование специального оператора выбора `switch`. Формат оператора следующий:

```
switch (выражение)
     {   [объявление]
               :
           [ case  константное-выражение1]: [ список-операторов1]
           [ case  константное-выражение2]: [ список-операторов2]
              :
              :
           [ default: [ список операторов ]]
     }
```

Выражение, следующее за ключевым словом `switch` в круглых скобках, может быть любым выражением, допустимыми в языке C/C++, значение которого должно быть целым. Значение этого выражения является ключевым для выбора из нескольких вариантов. Тело оператора `switch` состоит из нескольких операторов, помеченных ключевым словом `case` с последующим константным выражением. Обычно в качестве константного выражения используются целые или символьные константы (не может содержать переменные или вызовы функций). Все константные выражения в операторе `switch` должны быть уникальны. Кроме операторов, помеченных ключевым словом `case`, может быть, но обязательно один, фрагмент помеченный ключевым словом `default`.

Список операторов может быть пустым, либо содержать один

или более операторов. В операторе switch не требуется заключать последовательность операторов в фигурные скобки.

Можно использовать свои локальные переменные, объявления которых находятся перед первым ключевым словом case, однако в объявлениях не должна использоваться инициализация.

Схема выполнения оператора switch следующая: вычисляется выражение в круглых скобках; вычисленные значения последовательно сравниваются с константными выражениями, следующими за ключевыми словами case, если одно из константных выражений совпадает со значением выражения, то управление передается на оператор, помеченный соответствующим ключевым словом case, если ни одно из константных выражений не равно выражению, то управление передается на оператор, помеченный ключевым словом default, а в случае его отсутствия управление передается на следующий после switch оператор. Пример:

```
int i=2;
switch (i)
        {
         case 1: i += 2;
         case 2: i *= 3;
         case 0: i /= 2;
         case 4: i -= 5;
         default:      ;
       }
```

Рассмотрим ранее приведенный пример, в котором иллюстрировалось использование вложенных операторов if, переписанной теперь с использованием оператора switch:

```
char ZNAC;
int x,y,z;
switch (ZNAC)
     {
          case '+':  x = y + z;    break;
          case '-':  x = y - z;    break;
          case '*':  x = y * z;    break;
          case '/':  x = u / z;    break;
          default : ;
     }
```

Использование оператора break позволяет в необходимый момент прервать последовательность выполняемых операторов в теле оператора switch, путем передачи управления оператору, следующему за switch.

В теле оператора switch можно использовать вложенные операторы switch, при этом в ключевых словах case можно использовать

одинаковые константные выражения:

```
:
switch (a)
      {
        case 1: b=c; break;
        case 2:
            switch (d)
            {   case 0:  f=s;  break;
                case 1:  f=9;  break;
                case 2:  f-=9; break;
            }
         case 3: b-=c; break;
         :
      }
```

Задания для выполнения

Таблица 14

Выполнить задание, выделяя цифры числа, описанного стандартным числовым типом и используя условные операторы	
1	Дано четырехзначное натуральное *n*. Верно ли, что это число содержит ровно 2 одинаковые цифры?
2	Дано четырехзначное натуральное *n*. Верно ли, что это число содержит ровно 3 одинаковые цифры?
3	Дано натуральное *n*. Верно ли, что это все цифры числа различны?
4	Определить, равна ли сумма двух первых цифр заданного четырехзначного числа, сумме двух его последних цифр.
5	Определить, равен ли квадрат заданного трехзначного числа, кубу суммы цифр этого числа.
6	Определить, есть ли среди первых 3 цифр дробной части заданного положительного вещественного числа, цифра 0.
7	Определить, есть ли среди первых 5 цифр в дробной части заданного положительного вещественного числа, цифра 9.
8	Дано натуральное *n*. Является ли это число палиндромом учетом четырех цифр?
Выполнить задание, используя условные операторы	
9	Даны числа *a*, *b*, *c*, *d*. Если *a*<=*b*<=*c*<=*d*, то каждое число заменить наибольшим, если *a*>*b*>*c*>*d*, то числа - без изменений, в противном случае все числа заменить их квадратами.
10	Даны *x*, *y*, *z* вещественные числа. Существует ли треугольник с длинами сторон *x*, *y*, *z*? Если существует, то ответить – является ли он остроугольным.
11	Даны *a*, *b*, *c*, *d* – вещественные числа. Выяснить, можно ли прямоугольник со сторонами *a*, *b* поместить внутри прямоугольника со сторонами *c*, *d* так, чтобы каждая из сторон одного была параллельна или перпендикулярна каждой стороне другого.

12	Даны вещественные a_1 , b_1 , c_1 , a_2, b_2, c_2. Найти все решения системы линейных уравнений: $a_1x + b_1y+c_1 =0$ $a_2x+ b_2y +c_2 =0$
13	Даны числа a_1 , b_1 , c_1 , a_2, b_2, c_2. Напечатать координаты точки пересечения прямых, описываемых уравнениями $a_1x+b_1y=c_1$ и $a_2x+b_2y=c_2$, либо сообщить, прямые совпадают, не пересекаются, не существуют.
14	Даны $x_1, x_2, x_3, y_1, y_2, y_3$. Принадлежит ли начало координат треугольнику с вершинами $(x_1,y_1), (x_2 ,y_2), (x_3 ,y_3)$.
Выполнить задание, используя минимальное количество условных операторов	
15	Даны действительные числа x, y, z. Вычислить $max(x, y, z) * min(x, y, z)$.
16	Даны действительные числа x, y, z. Вычислить $min^2(x+y+z/2, xyz)+max(x, y)$.
17	Даны действительные числа x, y, z. Поменять значения переменных так, чтобы $x<y<z$.
18	Даны действительные числа x, y, z. Поменять значения переменных так, чтобы $x>=y>=z$.
19	Даны действительные числа x_1, y_1, x_2, y_2. Вычислить $max(x_1, y_1, x_2 ,y_2) * min(x_1, y_1, x_2, y_2)$.
20	Даны действительные числа x_1, y_1, x_2, y_2. Вычислить $max(x_1+y_1 ,x_1y_1, x_2+y_2, x_2y_2)+min(x_1, x_2, y_1, y_2)$.
21	Даны действительные числа x_1, y_1, x_2, y_2. Вычислить $min^2(x_1+y_1+x_2y_2,x_1y_1+x_2+y_2,x_1y_1+x_2y_2)+min(x_1y_1,x_2,y_2)$.
Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит 8: первое – номер вертикали, второе – номер горизонтали. Заданы натуральные числа *k, l, m, n*	
22	Выяснить являются ли поля (k,l) и (m,n) полями одного цвета.
23	На поле (k,l) расположена пешка. Угрожает ли она полю (m,n)?
24	На поле (k,l) расположена ладья. Угрожает ли она полю (m,n)?
25	На поле (k,l) расположен слон. Угрожает ли он полю (m,n)?
26	На поле (k,l) расположен король. Угрожает ли он полю (m,n)?
27	На поле (k,l) расположен конь. Угрожает ли он полю (m,n)?
28	Можно ли с поля (k,l) одним ходом ладьи попасть на поле (m,n)? Если нет, то выяснить, как это можно сделать за два хода.
29	Можно ли с поля (k,l) одним ходом слона попасть на поле (m,n)? Если нет, то выяснить, как это можно сделать за два хода.
30	Можно ли с поля (k,l) одним ходом короля попасть на поле (m,n)? Если нет, то выяснить, как это можно сделать за два хода
31	Можно ли с поля (k,l) одним ходом ферзя попасть на поле (m,n)? Если нет, то выяснить, как это можно сделать за два хода.

Продолжение таблицы 14

32	Можно ли с поля *(k,l)* одним ходом коня попасть на поле *(m,n)*? Если нет, то выяснить, как это можно сделать за два хода.
Выполнить задание, используя условные операторы	
33	Можно ли на прямоугольном участке застройки размером *a* на *b* метров разместить два дома размером в плане *p* на *q* и *r* на *s* метров? Дома можно располагать только параллельно сторонам участка.
34	Может ли шар радиуса *r* пройти через ромбообразное отверстие с диагоналями *p* и *q*?
35	Можно ли коробку размером *a*b*c* упаковать в посылку размером *r*s*t*? «Углом» укладывать нельзя.
36	Можно ли из круглой заготовки радиуса *r* вырезать две прямоугольные пластинки с размерами *a*b* и *c*d*?
37	Пройдет ли кирпич со сторонами *a, b* и *c* сквозь прямоугольное отверстие со сторонами *r* и *s*? Стороны отверстия должны быть параллельны граням кирпича.
38	Для заданного $0 \le n \le 200$, рассматриваемого как возраст человека, вывести фразу вида: «Мне 21 год», «Мне 32 года», «Мне 12 лет».
39	Лежит ли точка $M(x,y)$ внутри треугольника, заданного координатами своих вершин $A(x_A,y_A)$, $B(x_B,y_B)$, $C(x_C,y_C)$?
40	Заданы три числа: *a, b, c*. Определить, могут ли они быть сторонами треугольника, и если да, то определить его тип: равносторонний, равнобедренный, разносторонний. Замечание: условия существования треугольника: $a \le b+c$; $b \le a+c$; $c \le a+b$. Нельзя исключать экстремальных случаев, когда одна (или несколько) сторон равны нулю либо когда одно из неравенств переходит в равенство (треугольник нулевой площади).
41	Заданы координаты вершин треугольника ABC на плоскости. Вывести их в порядке обхода по часовой стрелке. Замечание: для проверки достаточно рассмотреть знаки внутренних углов.
42	Заданы три положительных числа A, B, C. Определить являются ли они сторонами равнобедренного треугольника.
43	Треугольник задан координатами своих вершин на плоскости: $A(x_a , y_a)$, $B(x_b,y_b)$, $C(x_c, y_c)$. Определить, является он прямо-, остро- или тупоугольным. Замечания: следует рассмотреть экстремальные случаи, когда вершины треугольника совпадают или лежат на одной прямой.
44	Четырехугольник ABCD задан координатами своих вершин на плоскости: $A(x_a,y_a)$, $B(x_b,y_b)$, $C(x_c,y_c)$, $D(x_d,y_d)$. Проверить, является ли он выпуклым. Замечани: есть несколько способов проверки выпуклости: анализ линейных неравенств, задаваемых сторонами; разбиение четырехугольника на треугольники со сравнением сумм их площадей и др.

Продолжение таблицы 14

45	Четырехугольник ABCD задан координатами своих вершин на плоскости: $A(x_a,y_a)$, $B(x_b,y_b)$, $C(x_c,y_c)$, $D(x_d,y_d)$. Определить тип четырехугольника: прямоугольник, параллелограмм, трапеция, произвольный четырехугольник. Учесть погрешность вычислений. Замечание: для устранения дополнительных источников погрешности рекомендуется использовать аппарат векторной алгебры: коллинеарность, равенство и ортогональность векторов – сторон четырехугольника
46	Проверить, лежит ли окружность $(x-a_1)^2+(y-b_1)^2=r_1^2$ целиком внутри окружности $(x-a_2)^2+(y-b_2)^2=r_2^2$ или наоборот.
47	Два отрезка на плоскости заданы координатами своих концов. Определить, имеют ли эти отрезки общие точки. Замечание: необходимо рассмотреть различные случаи взаимной ориентации отрезков: на одной прямой, на параллельных или пересекающихся прямых.
48	На шахматной доске стоят черный король и три белые ладьи (ладья бьет по горизонтали и вертикали). Проверить, не находится ли король под боем, а если есть угроза, то от кого именно.
49	На шахматной доске стоят черный король и белые ладья и слон (ладья бьет по горизонтали и вертикали, слон — по диагоналям). Проверить, есть ли угроза королю и если есть, то от кого именно.
50	На шахматной доске стоят три ферзя (ферзь бьет по вертикали, горизонтали и диагоналям). Найти те пары из них, которые угрожают друг другу.
51	Банк предлагает 3 вида срочных вкладов: на 3 месяца под *p1%*, на 6 месяцев под *p2%* и на год под *p3 %*. Какой из вкладов наиболее выгоден для вкладчика?
52	Можно ехать на такси со скоростью *v1* км/ч и оплатой *p1* р/км либо идти пешком со скоростью *v2* км/ч бесплатно. Как с наименьшими затратами преодолеть путь *s* за время *t*, если это возможно? Каковы эти затраты? Замечание: рекомендуется рассмотреть «запредельные» случаи: когда времени слишком мало, чтобы успеть даже на такси, либо слишком много, так что и пешком можно с запасом успеть до отхода поезда.
53	Из пункта А в пункт В выехал велосипедист со скоростью v_0 км/час. Одновременно навстречу ему из пункта В двинулся «автостопом» другой путник. s_1 м он двигался со скоростью v_1 км/час, s_2 м – со скоростью v_2 км/час, s_3 м – со скоростью v_3 км/час. Через сколько часов после старта и в какой точке путники встретились?
54	В ВУЗе принято, что старшая цифра номера студенческой группы означает номер факультета, средняя – последнюю цифру года поступления, младшая – порядковый номер группы на курсе. Продолжительность обучения – не более 6 лет (магистратура). Дан номер группы студента и текущий год. Напечатать, в каком году он поступил и на каком факультете учится. Пример: гр. 432, 1996 г. – факультет математический, год поступления 1993. Замечание: предусмотреть невозможные ситуации, например, гр. 521, год 2001.

Окончание таблицы 14

55	Имеются три раствора полезного вещества с концентрациями p_1, p_2 и p_3 каждый и стоимостью s_1, s_3 и s_3 соответственно. Можно ли смешать их так, чтобы получить раствор с заданной концентрацией p наименьшей стоимости? Указания: пусть a_1, a_2, a_3 — долевые содержания растворов в смеси. Тогда для получения заданной концентрации р необходимо: $p_1a_1+ p_2a_2+ p_3a_3=p$ Кроме того, нужно учесть условие «комплектности» смеси: $a_1+a_2+a_3=1$; $a_1\geq0$ $a_2\geq0$ $a_3\geq0$. При этих условиях необходимо найти наименьшее значение линейной функции: $s = s_1a_1+ s_2a_2+ s_3a_3$—> *min C* учетом ограничений задача сводится к минимизации линейной функции одного переменного на отрезке, однако искомые выражения и условия получаются достаточно громоздкими. Можно показать, что в решении будут участвовать не более двух растворов. Тогда достаточно среди вариантов: a) *a1=0* b) *a2=0* c) *a3=0* выбрать оптимальный, и затем провести необходимые расчеты.
56	Суточный рацион коровы составляет *b* кг сена, *v* кг силоса и *w* кг комбикорма. В хозяйстве, содержащем стадо из *k* голов, осталось *s* центнеров сена, *t* тонн силоса и *f* мешков комбикорма по 50 кг. Сколько еще дней хозяйство сможет кормить коров по полному рациону? Какой из кормов кончится раньше других?
57	Как известно, число делится на 3 тогда и только тогда, когда сумма его цифр делится на 3. Проверить этот признак на примере заданного трехзначного числа. Замечание: теоретическое утверждение о признаке делимости предлагается проверить на примере любого вводимого числа.
58	Среди заданных целых чисел *k, f, t* найти пары кратных.
59	Определить есть ли среди заданных целых чисел *A, B, C, D* хотя бы два чётных.
60	Проверить, является ли дробь *A/B* правильной.

Контрольные вопросы

1. В чем различие между символами = и ==?

2. Чем отличается переключатель `switch` от инструкции `if`?

3. Как можно использовать числа с плавающей точкой в конструкциях `switch`?

4. Объясните, как можно использовать инструкцию `if` для проверки правильности ввода?

5. Верно ли что оператор `break` производит выход только из ветвления `switch` наибольшей глубины вложенности?

Практикум № 5-6. ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ОПЕРАТОРЫ ПЕРЕХОДОВ

Краткие теоретические сведения

- Оператор цикла for

Оператор `for` – это наиболее общий способ организации цикла. Он имеет следующий формат:

```
for ( выражение 1 ; выражение 2 ; выражение 3 ) тело
```

Выражение 1 обычно используется для установления начального значения переменных, управляющих циклом. Выражение 2 – это выражение, определяющее условие, при котором тело цикла будет выполняться. Выражение 3 определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла.

Схема выполнения оператора for: вычисляется выражение 1; вычисляется выражение 2; если значения выражения 2 отлично от нуля (истина), выполняется тело цикла, вычисляется выражение 3 и осуществляется переход к пункту 2; если выражение 2 равно нулю (ложь), то управление передается на оператор, следующий за оператором `for`.

Проверка условия всегда выполняется в начале цикла. Это значит, что тело цикла может ни разу не выполниться, если условие выполнения сразу будет ложным. Пример:

```
int i,b;
 for (i=1; i<10; i++) b=i*i;   // вычисление квадратов чисел от 1 до 9
```

Некоторые варианты использования оператора `for` повышают его гибкость за счет возможности использования нескольких переменных, управляющих циклом:

```
int top,  bot;
char string[100],  temp;
        // для управления циклом используются две переменные top и bot
   for ( top=0, bot=100 ; top < bot ; top++, bot--)
      {  temp=string[top];
         string[bot]=temp;
      }
```

Другим вариантом использования оператора `for` является бесконечный цикл. Для организации такого цикла можно использовать пустое условное выражение, а для выхода из цикла обычно используют дополнительное условие и оператор `break`:

```
for (;;)
```

```
{ ...
  ...  break;
  ...
}
```

Так как согласно синтаксису языка оператор может быть пустым, тело оператора `for` также может быть пустым. Такая форма оператора может быть использована для организации поиска:

```
for (i=0; t[i]<10 ; i++) ;
//i – номер первого элемента массива t, значение которого больше 10
```

- Оператор цикла while

Оператор цикла `while` называется циклом с предусловием и имеет следующий формат:

```
while (выражение) тело ;
```

В качестве выражения допускается использовать любое выражение языка C/C++, в качестве тела любой оператор, в том числе пустой или составной.

Схема выполнения оператора `while` следующая: вычисляется выражение; если выражение ложно, то выполнение оператора `while` заканчивается и выполняется следующий по порядку оператор; если выражение истинно, то выполняется тело оператора `while`; процесс повторяется сначала.

Внутри операторов `for` и `while` можно использовать локальные переменные, которые должны быть объявлены с определением соответствующих типов.

- Оператор цикла do while

Оператор цикла `do while` называется оператором цикла с постусловием и используется в тех случаях, когда необходимо выполнить тело цикла хотя бы один раз. Формат оператора имеет следующий вид:

```
do тело while (выражение);
```

Схема выполнения оператора `do while`: выполняется тело цикла (которое может быть составным оператором); вычисляется выражение; если выражение ложно, то выполнение оператора `do while` заканчивается и выполняется следующий по порядку оператор; если выражение истинно, то выполнение оператора продолжается сначала.

Чтобы прервать выполнение цикла до того, как условие станет ложным, можно использовать оператор `break`.

Операторы `while` и `do while` могут быть вложенными:

```
int i,j,k;
  ...
  i=0;  j=0;  k=0;
  do { i++;
       j--;
       while (a[k] < i)  k++;
     }
  while (i<30 && j<-30);
```

- Оператор перехода break

Оператор `break` прерывает цикл. Его целесообразно использовать, когда условие продолжения итераций надо проверять в середине цикла:

```
// ищет сумму чисел до тех пор, пока не будет введено 100 чисел или 0
for(s=0, i=1; i<100;i++)
   {cin>>x;
    if( x==0) break; // если ввели 0, то суммирование заканчивается
    s+=x;
   }
```

- Оператор перехода continue

Оператор `continue`, как и оператор `break`, используется внутри операторов цикла, но в отличие от него выполнение программы продолжается не с оператора, следующего за прерванным оператором, а с начала прерванного оператора. Формат оператора следующий:

```
continue;
```

Пример:

```
int a,b;
  for (a=1,b=0; a<100; b+=a,a++)
   {if (b%2) continue;
               //передача управление на очередную итерацию цикла for
               //не выполняя операторы обработки четных сумм
     ... /* обработка четных сумм */
   }
```

Оператор `continue`, как и оператор `break`, прерывает самый внутренний из объемлющих его циклов.

- Оператор перехода goto

Использование оператора безусловного перехода `goto` в практике программирования C/C++ настоятельно не рекомендуется, так как он затрудняет понимание программ и возможность её модификации. Формат оператора следующий:

```
goto имя-метки;
       ...
имя-метки: оператор;
```

Оператор `goto` передает управление на оператор, помеченный меткой `имя-метки`. Помеченный оператор должен находиться в той же функции, что и оператор `goto`, а используемая метка должна быть уникальной, т.е. одно `имя-метки` не может быть использовано для разных операторов программы.

Любой оператор в составном операторе может иметь свою метку. Используя оператор `goto`, можно передавать управление внутрь составного оператора.

- Оператор return

Оператор `return` завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию, в точку, непосредственно следующую за вызовом. Функция `main` передает управление операционной системе. Формат оператора:

```
return [выражение];
```

Значение выражения, если оно задано, возвращается в вызывающую функцию в качестве значения вызываемой функции. Если выражение опущено, то возвращаемое значение не определено. Если функция не должна иметь возвращаемого значения, то ее нужно объявлять с типом `void`.

Пример:

```
int sum (int a, int b)
  {   renurn (a+b);    }
```

Задания для выполнения

Таблица 15

Найти сумму ряда с точностью ε=10^{-4}, общий член которого (при определении суммы членов ряда следует использовать рекуррентную формулу для получения следующего члена ряда, выводимую из отношения a_{n+1}/a_n ; что точность достигнута, если $a_n < \varepsilon$)			
1	$a_n = \frac{(-1)^{n-1}}{n^n}$	2	$a_n = \frac{2^n n!}{n^n}$
3	$a_n = \frac{1}{2^n} + \frac{1}{3^n}$	4	$a_n = \frac{3^n n!}{(3n)!}$
5	$a_n = \frac{1}{((3n-2)(3n+1))}$	6	$a_n = \frac{n!}{3n^n}$
7	$a_n = \frac{(2n-1)}{2^n}$	8	$a_n = \frac{(n!)^2}{2^{n^2}}$

9	$a_n = \frac{10^n}{n!}$	10	$a_n = \lg(n!)e^{-n\sqrt{n}}$
Найти сумму 13 членов ряда, в котором (привести варианты решения этой задачи с использованием итерационных циклов for, while, do..while)			
11	$a_n = \frac{\ln(n!)}{n^2}$	12	$a_n = \frac{n!}{n^{\sqrt{n}}}$
13	$a_n = \frac{n^{\ln n}}{(\ln n)^n}$	14	$a_n = e^{-\sqrt{n}}$
Записать программу табулирования на отрезке условной функции в 2-х вариантах (с циклами while, for). Вывод результатов на экран обеспечить в виде таблицы с "шапкой"			
15	$y = \begin{cases} at^2 \ln t, & 1 \le t \le 2 \\ 1, & t < 1 \\ e^{\alpha t}\cos bt, & t > 2 \end{cases}$ a= - 0.5, b=2, $t \in [0;3]$, Δt=0.15	16	$y = \begin{cases} \pi x^2 - \frac{7}{x^2}, & x < 1.3 \\ ax^3 + 7\sqrt{x}, & x = 1.3 \\ \lg(x + 7\sqrt{x}, & x > 1.3 \end{cases}$ a= 1.5, $x \in [0.8;2]$, Δx=0.15
17	$\omega = \begin{cases} ax^2 + bx + c, & x < 1.2 \\ a/x + \sqrt{x^2+1}, & x = 1.2 \\ (a+bx)/\sqrt{x^2+1}, & x > 1.2 \end{cases}$ a= 2.8, b=-0.3, c=4, $x \in [1;2]$, Δx=0.05	18	$Q = \begin{cases} \pi x^2 - 7/x^2, & x < 1.4 \\ ax^3 + 7\sqrt{x}, & x = 1.4 \\ \ln(x + 7\sqrt{\lvert x+a \rvert}, & x > 1.4 \end{cases}$ a= 1.65, $x \in [0;3]$, Δx=0.15
Составить программу табулирования на отрезке с шагом 0.01 функции, имеющей вид многочлена. Значение многочлена вычислять по схеме Горнера. Вывод результатов обеспечить в виде таблицы			
19	$y = 2x^9 + 3x^7 + x^5 + x^3 - 2x + 1$ x ∈ [0; 1]	20	$z = x^{10} + 3.7x^8 + x^7 - x^6 - 0.5x^4 - x^2$ x ∈ [1; 1.5]
21	$y = 12x^9 + 3x^7 + x^5 + x^3 - 20x$ x ∈ [0; 1]	22	$z = 1 + x + \frac{x^2}{2!} + \ldots + \frac{x^{10}}{10!}$ x ∈ [1; 2]
Дано натуральное n. Вычислить сумму ряда двумя способами – организовав вычисления суммы слева направо и справа налево. Сравнить и объяснить полученные результаты			
23	$\sum_{k=1}^{n} \frac{1}{k \cdot (k+1) \cdot \ldots \cdot k^2}$	24	$\sum_{k=1}^{n} \frac{1}{(k^2)!}$

Продолжение таблицы 15

25	$\sum_{k=1}^{n} \frac{(2\cdot k)!+x^k}{(k^2)!}$	26	$\sum_{k=1}^{n} k^k \cdot 1/x^{2\cdot k}$
27	$\sum_{k=1}^{n} \frac{\sum_{m=1}^{k} \sin(k\cdot m)}{k!}$	28	$\sum_{k=1}^{n} \frac{1/2+1/3+...+1/(k+1)}{k^k}$
Выполнить следующее задание без хранения последовательностей значений			
29	Дано *n* вещественных чисел (*n* заранее не известно). Найти порядковый номер того из них, которое наиболее близко к какому-либо целому.		
30	Дано *n* вещественных чисел (*n* заранее не известно). Определить, сколько из них больше своих соседей, т.е. предыдущего и последующего.		
31	Дана непустая последовательность ненулевых целых чисел, за которой следует 0. Определить, сколько раз в этой последовательности меняется знак.		
32	Дано *n* целых чисел (*n* заранее не известно). Определить количество чисел в наиболее длинной подпоследовательности из подряд идущих нулей.		
Выполнить задание, используя перебор значений			
33	Определить количество трехзначных, натуральных чисел, сумма цифр которых равна *n*.		
34	Вывести на экран в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр.		
35	Дано натуральное число *n*. Получить все такие натуральные *q*, что n делится на q_2 и не делится на q_3.		
36	Дано натуральное *n*. Указать *x, y, z* таких натуральных чисел, что $n=x^2+y^2+z^2$ и $x \le y \le z$.		
37	Даны натуральные числа *m, n*. Получить все меньшие *n* натуральные числа, квадрат суммы цифр которых равен *m*.		
38	Для заданного натурального *N* определить наименьшее число *S*, которое можно представить в виде суммы *aN* + *bN* по крайней мере двумя различными способами (*a, b* – натуральные числа, представления, отличающиеся лишь порядком слагаемых, различными не считаются).		
39	Найти все натуральные числа, не превосходящие заданного *N* и делящиеся на каждую из своих цифр.		
40	Найти все пары двухзначных натуральных чисел *M, N* таких, что значение произведения *M*N* не изменится, если поменять местами цифры каждого из сомножителей (такой парой будет, например, 38 и 83).		
41	Найти минимальное число, которое представляется суммой четырех квадратов натуральных чисел не единственным образом.		
42	Ввести с клавиатуры натуральное число *n*. Определить все способы выплаты суммы *n* с помощью купюр достоинством 1, 5, 10, 20 и 100. Определить способ выплаты суммы *n* с помощью наименьшего числа купюр.		

Продолжение таблицы 15

43	Два двузначных числа, записанных подряд, образуют четырехзначное число, которое делится на их произведение. Найти эти числа.
44	Заданы три натуральных числа *A, B* и *N*. Найти все натуральные числа, не превосходящие *N*, которые можно представить в виде суммы (произвольного числа) слагаемых, каждое, из которых – *A* или *B*.
45	Найти все такие целые *a, b*, что $n=3a+5b$ для любого натурального $n>7$.
46	Найти все числа из диапазона от *n* до *m*, которые при возведении в квадрат дают палиндром.
47	Найти все натуральные числа, не превосходящие заданного *N*, десятичная запись которых есть строго возрастающая или строго убывающая последовательность цифр.
48	Найти сумму целых положительных чисел, кратных 3 и меньших 200.
49	Найти сумму целых положительных чисел, больших 20, меньших 100 и кратных 3
Выполнить задание, используя алгоритмы нахождения делителей числа	
50	Дано натуральное число *n*. Получить все его натуральные делители.
51	Даны натуральные числа *n, m*. Получить их общие делители (<0 и >0).
52	Даны натуральные числа *n, m*. Получить все их общие кратные, < *mn*.
53	Даны натуральные числа *n, m*. Получить *НОК(n,m)*. *(НОК(n,m)=nm/НОД(n,m))*.
54	Найти наибольший общий делитель для 10 заданных натуральных чисел.
55	Даны натуральные числа $n_1,n_2,...n_m$ $(m>=2)$. Вычислить $НОД(n_1,n_2,...n_m)$.
56	Найти натуральное число от *n* до *k* с максимальной суммой делителей.
57	Найти натуральное число из диапазона от *n* до *k*, которое имеет наибольшее количество делителей.
58	Найти все пары дружественных чисел от *n* до *k*. Два числа называются дружественными, если каждое из них равно сумме всех делителей другого, кроме самого этого числа.
59	Найти все совершенные числа, меньшие *n*. Число – совершенное, если оно равно сумме всех своих делителей, за исключением самого числа.
60	Построить первые *N* натуральных чисел, делителями которых являются только числа 2, 3 и 5.
Выполнить задание, используя алгоритмы для нахождения простых чисел	
61	Найти все простые числа, не превосходящие заданного $N>0$.
62	Дано натуральное *n*. Получить все натуральные числа меньшие *n* взаимно простые с ним.
63	Даны натуральные *p, q*. Получить все делители числа p взаимно простые с числом *q*.
64	Среди всех четырехзначных чисел получить все простые числа, каждое из которых обладает тем свойством, что сумма первых двух цифр равна сумме двух последних цифр.

Окончание таблицы 15

65	Дано натуральное число *n*. Получить его каноническое разложение (разложение на простые множители).
66	Дано натуральное число *n*. Выяснить, имеются ли среди чисел *n, n+1,…,2n* числа-близнецы, т.е. простые числа, разность между которыми равна 2.
67	Натуральное число, записанное в десятичной системе счисления, называется сверхпростым, если оно остается простым при любой перестановке своих цифр. Определить все сверхпростые числа до *n*.
Выполнить задание без хранения последовательностей. Дано натуральное *k*	
68	Определить *k*-ю цифру последовательности: 110100100010000..., в которой выписаны подряд степени 10.
69	Определить *k*-ю цифру последовательности: 1248163264..., в которой выписаны подряд степени 2.
70	от 1 до 180. Определить, какая цифра находится в *k*-ой позиции последовательности: 10111213...9899, в которой выписаны подряд все двузначные числа.
71	Вывести *k* цифру последовательности 12345678910111213...
72	Вывести *k* цифру последовательности (квадраты натуральных чисел) 149162536 ...
73	Вывести *k* цифру последовательности (числа Фибоначчи) 1123581321 ...
Составить программу решения задачи	
74	В учебном заведении задается начало учебного дня, продолжительность урока, продолжительность обычного и большого перерывов (и их «место» в расписании), количество уроков. Получить расписание звонков на весь учебный день.
75	Составить алгоритм решения ребуса РАДАР=(Р+А+Д)^4 (различные буквы означают различные цифры, старшая - не 0).
76	Составить алгоритм решения ребуса МУХА+МУХА+МУХА=СЛОН (различные буквы означают различные цифры, старшая - не 0).
77	Составить алгоритм решения ребуса ДРУГ-ГУРД=2727 (различные буквы означают различные цифры, старшая - не 0).
78	Составить алгоритм решения ребуса КОТ+КОТ=ТОК (различные буквы означают различные цифры, старшая - не 0).

Контрольные вопросы

1. Назовите и опишите основное назначение каждого из трех выражений, входящих в состав оператора цикла `for`.

2. Какими критериями вы будете руководствоваться при выборе цикла `for`, `do` или `while`?

3. Что такое вложенный цикл?

4. Каково назначение инструкции `break`?

5. Создайте цикл `for`, вывода на экран числа от 100 до 0.

Практикум №7. ОДНОМЕРНЫЕ МАССИВЫ. УКАЗАТЕЛИ

Краткие теоретические сведения

- Определение массива

Массив (или вектор) – это группа элементов одного типа (`double`, `float`, `int` и т.п.). Из объявления массива компилятор должен получить информацию о типе элементов массива и их количестве. Объявление массива имеет два формата:

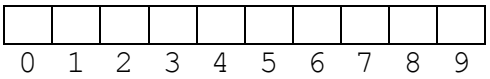
```
спецификатор-типа описатель [константное - выражение];
спецификатор-типа описатель [ ];
```

Описатель – это идентификатор массива. Спецификатор-типа задает тип элементов объявляемого массива. Элементами массива не могут быть функции и элементы типа `void`.

Константное-выражение в квадратных скобках задает количество элементов массива. Константное-выражение при объявлении массива может быть опущено в следующих случаях: при объявлении массив инициализируется; массив объявлен как формальный параметр функции; массив объявлен как ссылка на массив, явно определенный в другом файле.

Пример определения массива:

```
int arr[10];
```



Индексы элементов в массиве `arr` могут меняться от 0 до 9, всего в массиве 10 элементов.

- Инициализация массива

Инициализация массивов выполенеа программно, а также при определении:

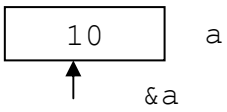
```
double d[] = {1, 2, 3, 4, 5};
```

Длина массива вычисляется компилятором по количеству значений перечисленных в фигурных скобках.

- Указатели и ссылки

Каждая переменная в программе это объект, имеющий имя и значение. По имени можно обратиться к переменной и получить ее значение. Оператор присваивания (=) выполняет обратное действие: имени переменной ставится в соответствие значение.

```
a=10;
```



Выражение `&a` позволяет получить адрес участка памяти, выделенного переменной `a` и означает ссылка на `a`. Операция `&` применима только к объектам имеющим имя и размещенным в памяти (переменным, массивам, структурам, строковым литералам и т.п.). Ни одна операция на ссылку не действует:

```
int i = 0;
int& r = i;
r++;        //не увеличивает ссылку, а i увеличивается на 1
```

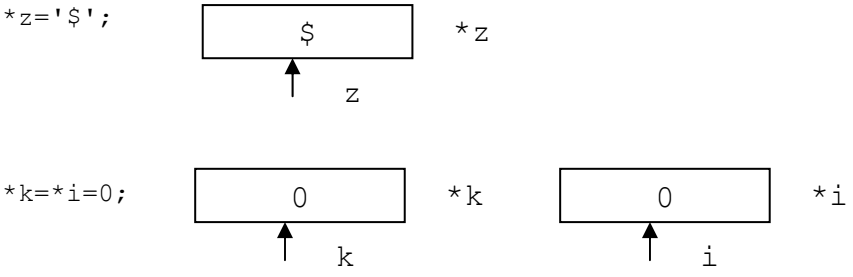
Имея возможность определить ссылку (адрес переменной) с помощью `&`, надо иметь возможность работать с ней: сохранять, передавать, преобразовывать. Для этого вводится понятие указателя. Указатель – это переменная, значением которой служит адрес объекта конкретного типа. Формат объявления указателя:

```
спецификатор-типа *описатель
```

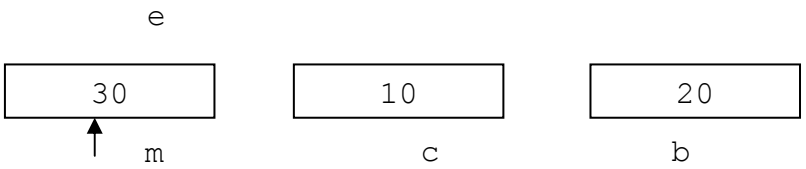
Спецификатор-типа задает тип объекта и может быть любого основного типа, типа структуры, смеси (об этом будет сказано ниже) или `void`. Чтобы выполнить арифметические и логические операции над указателями или над объектами, на которые они указывают, необходимо при выполнении каждой операции явно определить тип объектов. Такие определения типов может быть выполнено с помощью операции приведения типов. Пример:

```
char *z;
int *k,*i;
float *f;
```

Здесь * – это операция разыменования, то есть ссылка на объект, на который указывает указатель. Операндом этой операции всегда является указатель. Эта операция также называется косвенным обращением. Результат операции – это тот объект, который адресует указатель-операнд.



```
int e, c, b, *m;
m = &e ;    // в m хранится адрес e
*m  =  c  +  b  ;
```

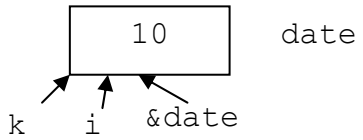


- Операции над указателями

Над указателями возможны опреации: присваивание (=) указателей одного типа; получение значения объекта, на который ссылается указатель (*); получение адреса самого указателя (&).

Пример:

```
int date = 10;
int *i, *k;;
i = &date;
k = i;
z = NULL;
```



Допустимы также операции сравнения: ==, !=, <, <=, >, >=, адресной арифметики и все производные от нее: +, -, ++, --. С помощью унарных операций ++ и -- числовые значения переменных типа указатель меняются по разному, в зависимости от типа данных, с которым связаны эти переменные:

```
char *z;
int *k,*i;
float *f;
z++; // значение указателя изменяется на 1
i++; // значение указателя изменяется на 2
f++; // значение указателя изменяется на 4
```

При изменении указателя на 1, указатель переходит к началу следующего (предыдущего) поля той длины, которая определяется типом объекта, адресуемого указателем.

Снабжение описания указателя префиксом const делает объект, но не сам указатель, константой:

```
const char* pc = "asdf";     // указатель на константу
pc[3] = 'a';                 // ошибка
pc = "ghjk";                 // ok
```

Чтобы описать сам указатель, а не указываемый объект, как константный, используется операция const*:

```
char *const cp = "asdf";     // константный указатель
cp[3] = 'a';                 // ошибки нет
cp = "ghjk";                 // ошибка
```

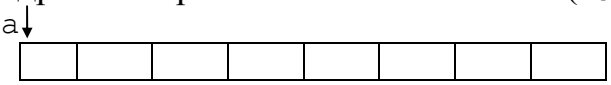
Чтобы сделать константами оба объекта, их оба нужно описать const:

```
const char *const cpc = "asdf";     // const указатель на const
cpc[3] = 'a';                       // ошибка
cpc = "ghjk";                       // ошибка
```

Указателю на константу можно присваивать адрес переменной. Однако нельзя присвоить адрес константы указателю, на который не было наложено ограничение, поскольку это позволило бы изменить значение объекта.

- Указатели и массивы

Имя массива без индекса является указателем-константой, т. е. адресом первого элемента массива (`a[0]`).




```
int a[8],*pa;
*a = = a[0] ;      //эквивалентно
*(a+1) = = a[1];   //эквивалентно
*(a+i) = =a[i];    //эквивалентно
pa=a+5;            //указатель на 5-ый элмеент массива
pa=&a[2];          //указатель на 2-ый элмеент массива
a[3]=*(pa+1);      //3-ему элементу массива присвоить значение 1-ого
```

Необходимо иметь в виду, что указатель является переменной, так что операции `pa=a` и `pa++` имеют смысл, но имя массива является константой, а не переменной поэтому следующие конструкции будут не допустимы:

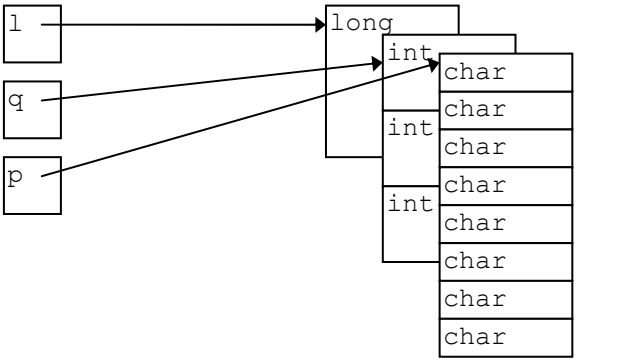
```
a=pa;       //ошибка
a++;        //ошибка
pa=&a;      //ошибка
```

- Присваивание указателей различного типа

Операцию присваивания указателей различных типов следует понимать как назначение указателя в левой части на ту же самую область памяти, на которую назначен указатель в правой. Оба указателя после присваивания содержат один и тот же адрес. Но поскольку тип указуемых переменных у них разный, то эта область памяти по правилам интерпретации указателя будет рассматриваться как заполненная переменными либо одного, либо другого типа:

```
char  *p, A[20];
int   *q;
```

```
long  *l;
p = A;        q = (int*)p; l = (long*)p;
```



Задания для выполнения

При выполнении работы используются статические массивы. Для организации статических массивов с псевдопеременными границами необходимо объявить массив достаточно большой длины. Затем пользователь вводит реальную длину массива (не больше *N*) и работает с массивом той длины, которую он сам указал. Остальные элементы (хотя память под них и будет выделена) не рассматриваются. При уменьшении или увеличении длины массива необходимо изменять его реальную длину.

Таблица 16

Сформировать одномерный массив целых чисел, используя датчик случайных чисел и выполнить задание 2-мя вариантами (с использованием индексов и указателей)	
1	Удалить элемент с номером *k*. Добавить после каждого четного элемента массива элемент со значением 0.
2	Удалить первый элемент равный 0. Добавить после каждого четного элемента массива элемент со значением *m[i-1]+2*.
3	Удалить все элементы равные 0. Добавить после первого четного элемента массива элемент со значением *m[i-1]+2*.
4	Удалить элементы, индексы которых кратны 3. Добавить после каждого отрицательного элемента массива элемент со значением | *m[i-1]+1*|.
5	Удалить элементы кратные 7. Добавить после каждого нечетного элемента массива элемент со значением 0.
6	Удалить последний элемент равный 0. Добавить после элемента массива с заданным индексом элемент со значением 100.
7	Удалить все элементы с заданным значением. Добавить перед каждым четным элементом массива элемент со значением 0.
8	Удалить первый элемент с заданным значением. Сдвинуть массив циклически на *k* элементов вправо.
9	Удалить 5 первых элементы массива. Добавить в конец массива 3 новых элемента.

Продолжение таблицы 16

10	Поменять местами минимальный и максимальный элементы массива. Удалить из массива все элементы превышающие его среднее значение более, чем на 10%.
11	Удалить из массива все элементы совпадающие с его минимальным значением. Добавить в начало массива 3 элемента с значением равным среднему арифметическому массива.
12	Перевернуть массив и, если число элементов массива нечетное, удалить его средний элемент. Добавить в начало массива 3 элемента с значением *m[i+10]-2*.
13	Найти в массиве и вывести значение наиболее часто встречающегося элемента.
14	Найти в массиве элемент, наиболее близкий к среднему арифметическому суммы его элементов.
15	Разделить массив на две части, поместив в первую элементы, большие среднего арифметического их суммы, а во вторую - меньшие (части не сортировать).
16	Найти наименьшее общее кратное всех элементов массива (то есть числа, которое делится на все элементы).
17	Найти наибольший общий делитель всех элементов массива.
18	Интервал между минимальным и максимальным значениями элементов массива разбить пополам и относительно этого значения разбить массив на две части (части не сортировать).
19	Найти в массиве наибольшее число подряд идущих одинаковых элементов (например {1,5,3,6,6,6,6,6,3,4,4,5,5,5} = 5).
Выполнить задание. Если в учловии не указано, считать что задан масив(ы) целых чисел	
20	По заданному массиву *A* размера *n* построить массив *B* такой, что *B(i)* – это количество элементов из *A*, не превосходящих *A(i)*, в конечном отрезке *A* длиной *i-1*.
21	Найти максимальную по длине монотонную (т. е. либо неубывающую, либо невозрастающую) подпоследовательность заданного массива целых чисел.
22	Определить, содержится ли в заданнм массиве хотя бы одно число Фибоначчи.
23	В массиве *C(m)* каждый третий элемент заменить полусуммой двух предыдущих, а стоящий перед ним — полусуммой соседних с ним элементов. Дополнительный (рабочий) массив не использовать.
24	Каждый из элементов *ti* массива *T(n)* заменить минимальным среди первых *i* элементов этого массива.
25	В массиве *A(n)* каждый элемент, кроме первого, заменить суммой всех предыдущих элементов.

Продолжение таблицы 16

26	Дан массив целых чисел *K(n)*. Найти в нем минимальный k_{min} и максимальный k_{max} элементы. Вывести в порядке возрастания все целые числа из интервала (k_{min},k_{max}), не встречающиеся в исходном массиве
27	Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами
28	Найти все числа, встречающиеся в массиве *P(m)* строго два раза (не упорядочивая самого массива).
29	В массиве *Z(m)* найти число чередований знака, то есть число переходов с минуса на плюс или с плюса на минус. Пример: в последовательности 0, -2, 0, -10, 2, -1, 0, 0, 3, 2, -3 четыре чередования (как известно, нуль не имеет знака).
30	Дан массив *A(n)*. Все положительные его элементы поместить в начало массива *B(n)*, а отрицательные элементы — в начало массива *C(n)*. Подсчитать количество тех и других.
31	В массиве *T(k)* найти первый и последний нулевые элементы.
32	Задан массив, определить значение k, при котором сумма $\|A(1)+A(2)+...A(k)-A(k+1)+...+A(N)\|$ минимальна (то есть минимален модуль разности сумм элементов в правой и левой части, на которые массив делится этим *k*).
33	В массиве *M(k)* много совпадающих элементов. Найти количество различных элементов в нем (не упорядочивая массива).
34	Даны два массива A и B размера *m*, элементы которых упорядочены по возрастанию. Объединить эти массивы так, чтобы результирующий массив остался упорядоченным.
35	В целочисленном массиве *K(n)* много повторяющихся элементов. Найти (в процентах) частоту появления каждого из *m* наиболее часто встречающихся элементов ($m<<n$).
36	Даны два целочисленных массива *K(m)* и *L(n)*. Найти наибольший элемент массива K, не имеющий себе равных в массиве *L*.
37	Из элементов массива *A(2n)* получить массивы *B(n)* и *C(n)* следующим образом. Выбрать в массиве *A* два наиболее близких по значению элемента; меньший из них поместить в массив *B*, а больший — в *C*, Продолжить выбор из оставшихся элементов до полного заполнения массивов *B* и *C*.
38	Все четные элементы целочисленного массива *K(n)* поместить в массив *L(n)*, а нечетные — в массив *M(n)*. Подсчитать количество тех и других.
39	Целочисленный массив *K(n, n)* заполнить нулями и единицами, расположив их в шахматном порядке.
40	В массиве *Z(n)* найти наиболее длинную цепочку стоящих подряд попарно различных элементов
41	В массиве *A(2n+1)*, не содержащем одинаковых элементов, найти средний по величине элемент, то есть такой, что в массиве *A* ровно *n* элементов

	меньше его и столько же элементов больше его. Массив *A* сохранить (не сортировать), дополнительных массивов не использовать

Окончание таблицы 16

42	Последовательность $a_1,a_2,...,a_k$, называется пилообразной, если $a_1<a_2>a_3<a_4>...>a_k$ либо $a_1>a_2<a_3>a_4<...<a_k$. В массиве *A(n)* найти самую длинную пилообразную последовательность
43	Дано вещественное число *R* и массив размера *N*. Найти два элемента массива, сумма которых наиболее и наименее близка к данному числу.
44	Имеются результаты *n* ежедневных измерений количества выпавших осадков. За какую из недель (отрезок времени длиной 7 дней), считая с начала периода измерений, выпало наибольшее количество осадков?
45	В массиве *K(n)* в порядке убывания представлены достоинства денежных знаков (купюр и монет) валютной системы некоторой страны. Реализовать выдачу в этой системе заданной суммы *m* минимальным числом денежных знаков.
46	Сформировать массив простых множителей заданного числа.
47	Известно, что 1 января 1999 г. – пятница. Для любой заданной даты программа должна выводить день недели.
48	Известно, что 1 января 1999 г. – пятница. Программа должна найти все “черные вторники” и “черные пятницы” 1999 года (то есть – 13 числа).
49	Известно время $t_1,t_2,...,t_n$, за которое некоторую работу может выполнить каждый из n рабочих бригады, работая в одиночку. Сколько времени понадобится бригаде на выполнение этой работы, если они будут работать совместно.
50	Из пункта А в пункт В, между которыми 5 км, выехал велосипедист с постоянной скоростью V_0 км/ч. Навстречу ему – из пункта В – другой путешественник решил добраться «автостопом» — на разных видах попутного транспорта. Перед каждым участком он k_i минут «голосует», ожидая попутного транспорта, затем движется t_i часов со скоростью v_i км/час (величины k_i, t_i, v_i, $i=1,2,...,n$ заданы в соответствующих массивах).

Контрольные вопросы

1. Что такое массив?
2. Может ли массив содержать переменные нескольких типов?
3. Какова взаимосвязь между значением индекса и значением элемента массива?
4. Что такое указатель? Какие операции возможны над указателями?
5. Как связаны указатели и массивы?

Практикум №8. СТРОКИ

Краткие теоретические сведения

Для представления символьной информации можно использовать символы, символьные переменные и символьные константы.

Символьная константа представляется последовательностью символов, заключенной в кавычки: *“Начало строки \n”*. В C/C++ нет отдельного типа для строк. Массив символов – это и есть строка. Количества элементов в таком массиве на один элемент больше, чем изображение строки, т. к. в конец строки добавлен *‘\0’* (нулевой байт или нуль-терминатор).

символ(1 байт) строка (2 байта)

Поместить строку в массив можно либо при вводе, либо с помощью инициализаций:

```
char s[] = “STRING”;
char s[20] = { 'S','t','r','i','n','g','\0' };
```

В соответствии с общепринятым стандартом установлено соответствие между символами и кодами. Клавиатура (совместно с драйвером) кодирует нажатие любой клавиши с учетом регистровых и управляющих клавиш в соответствующий ей код. Например:

```
' '   - 0x20,          'B'   - 0x42,
'*'   - 0x2A,          'Y'   - 0x59,
```

Некоторые программы и стандартные функции обработки символов и строк используют тот факт, что цифры, прописные и строчные (маленькие и большие) латинские буквы имеют упорядоченные по возрастанию значения кодов:

```
'0' - '9'   0x30 - 0x39
'A' - 'Z'   0x41 - 0x5A
'a' - 'z'   0x61 - 0x7A
```

Текст является упорядоченным множеством строк, и представляется массивом символов:

```
char  A[20][80];
char  B[][40] = { "String","New string","0000","abcdef"};
```

Первый индекс двумерного массива соответствует номеру стро-

ки, второй – номеру символа в нем.

- Функции работы со строками

Для работы со строками существует специальная библиотека `string.h`. Функции из котороой приведены в табл. 17.

Таблица 17

Функция	Прототип и краткое описание функции
`strcmp`	`int strcmp(const char *str1, const char *str2);` Сравнивает строки str1 и str2. Если str1< str2, то результат отрицательный, если str1 = str2, то результат равен 0, если str1> str2, то результат положительный.
`strcpy`	`char* strcpy(char*s1, const char *s2);` Копирует байты из строки s1 в строку s2
`strdup`	`char *strdup (const char *str);` Выделяет память и перености в нее копию строки str.
`strlen`	`unsigned strlen (const char *str);` Вычисляет длину строки str.
`strncat`	`char *strncat(char *s1, const char *s2, int kol);` Приписывает kol символов строки s1 к строке s2.
`strncpy`	`char *strncpy(char *s1, const char *s2, int kol);` Копирует kol символов строки s1 в строку s2.
`strnset`	`char *strnset(char *str, int c, int kol);` Заменяет первые kol символов строки s1 символом c.
`atoi`	`int atoi(char *str);` Преобразует строку в целое
`atof`	`float atof(char *str);` Преобразует строку в число с плавающей точкой

Строки, при передаче в функцию, в качестве фактических параметров могут быть определены либо как одномерные массивы типа `char[]`, либо как указатели типа `char*`. В отличие от массивов, в этом случае нет необходимости явно указывать длину строки.

Для указателя с типом указуемой переменной `char` допускаются различные интерпретации: указатель на отдельный байт; указатель на область памяти – массив байтов; указатель на отдельный символ; указатель на массив символов.

Задания для выполнения

Таблица 18

Задана строка символов	
1	Определить количество символов в наиболее длинной упорядоченной по возрастанию последовательности символов.

Продолжение таблицы 18

2	Определить количество символов в наиболее длинной упорядоченной по убыванию последовательности символов.
3	Заменить в строке все вхождения “child” на “children”.
4	Исключить из строки группы символов, расположенные между (*...*) вместе со скобками. Предполагается, что нет вложенных скобок.
5	Удалить в строке все вхождения “abc”.
6	Удалить в строке все буквы “b”, непосредственно за которыми идет цифра.
7	Преобразовать строку следующим образом: после каждой буквы “й”добавить символ “!”
8	Преобразовать строку следующим образом: удалить все символы “*”, и повторить каждый символ, отличный от “*”.
9	В строке есть символ “*”. Получить все символы между первым и вторым символом“*”. Если второго символа “*” нет, то получить все символы после единственного символа “*”.
10	Определить число вхождений в строку подстроки “aba”.
11	Исключить из строки группы символов, расположенные между (...) вместе со скобками. Предполагается, что нет вложенных скобок.
Выполнить следующие задания, используя строковый тип	
12	Дан текст из заглавных латинских букв. Определить, является ли этот текст правильной записью римскими цифрами целого числа 1 до 1999, и если является, вывести на экран это число арабскими цифрами (в десятичной системе).
13	Заданное целое число от 1 до 1999 вывести римскими цифрами.
14	Из заданного текста выбрать и напечатать те символы, которые встречаются в нем ровно один раз (в том порядке, как они встречаются в тексте).
15	Из заданного текста выбрать и напечатать символы, которые встречаются в нем более одного раза (в том порядке, как они встречаются в тексте).
16	Проверить, имеется ли в заданном тексте баланс открывающих и закрывающих круглых скобок, т. е. верно ли, что можно установить взаимно однозначное соответствие открывающих и закрывающих скобок со следующими свойствами открывающая скобка всегда предшествует соответствующей закрывающей.
17	Для заданного текста определить длину содержащейся в нем максимальной серии символов, отличных от букв.
18	Для каждого символа заданного текста указать, сколько раз он встречается в тексте. Сообщение об одном символе должно печататься не более одного раза.
Выполнить следующие задания, не используя стандартные операции и процедуры для строк символов	
19	Написать программу, которая осуществляет сравнение двух строк.

Продолжение таблицы 18

20	Написать программу получения строки, в которой удалены все «лишние» пробелы, из нескольких подряд идущих пробелов оставить только один.
21	Написать программу, которая удаляет в строке *S* все символы, не являющиеся буквами или цифрами и заменяющая каждую большую букву одноименной малой (для латинских и русских букв).
22	Написать программу, реализующую удаление *k* символов с позиции номер n из строки *S.*
23	Написать программу, реализующую вставку подстроки *S1* длиной *k* в строку *S* с позиции номер *n*.
24	Написать программу, реализующую конкатенацию *k* строк.
25	Написать программу, реализующую выделение подстроки *S1* длиной *k* с позиции номер *n* из строки *S.*
26	Написать программу, реализующую определение номера позиции, с которой подстрока *S1* длиной *k* входит в строку *S*. Удалить в строке x все вхождения подстроки y.
27	Написать программу для определения номера позиции, с которой подстрока *ss* входит в часть строки *s,* начинающейся с *k* позиции. Удалить в строке x все вхождения подстроки y.
Задана строка символов. Группы символов, разделенные символами-разделителями (одним или несколькими) будем называть словом. Символы-разделители: пробел,.,;:?!-()	
28	Вывести слова, в которых заменить каждую большую букву одноименной малой; удалить все символы, не являющиеся буквами или цифрами; вывести в алфавитном порядке все гласные буквы, входящие в каждое слово строки.
29	Вывести текст, составленный из последних букв всех слов; все слова, содержащие ровно 2 буквы ‘d’.
30	Вывести слова строки начинающиеся и заканчивающиеся одной и той же буквой, содержат ровно три буквы ‘k’.
31	Найти количество слов в строке, заканчивающихся на буквы 'ть', вывести каждое слово строки в обратном порядке.
32	Вывести на экран слова, в которых только русские буквы; вывести в алфавитном порядке все согласные русские буквы, не входящие в строку.
33	Определить в этой строке слова, в которых нет одинаковых символов; вывести текст, удалив из него повторные вхождения слов.
34	Вывести слова, в которых нет повторяющихся букв; буквы упорядочены по алфавиту; с длиной максимальной в этой строке.
35	Определить слова, в которых текст упорядочен ли по алфавиту, вывести в алфавитном порядке все гласные, входящие в этот текст по одному разу.
36	Вывести слово, наиболее часто встречающееся в строке, последовательность слов, в которых буквы упорядочены по алфавиту.
37	Перечислить все слова заданного предложения, которые состоят из тех же букв, что и первое слово предложения.

Окончание таблицы 18

38	В заданном предложении найти пару слов, из которых одно является обращением другого (обращение слова – слово, получающееся из исходного записью его букв в обратном порядке).
39	Для каждого из слов заданного предложения указать, сколько раз оно встречается в предложении.
40	Найти самое длинное симметричное слово заданного предложения.
41	Расстояние между двумя словами равной длины – это количество позиций, в которых различаются эти слова. В заданном предложении найти пару наиболее далеко удаленных слов заданной длины.
42	Отредактировать заданное предложение, удаляя из него слова-серии, а также те слова, которые уже встречались в предложении раньше.
43	Найти множество всех слов, которые встречаются в каждом из двух заданных предложений.
44	Отредактировать заданное предложение, удаляя из него все слова с нечетными номерами и переворачивая слова с четными номерами. Пример: HOW DO YOU DO -> OD OD.
45	Найти самое длинное общее слово двух заданных предложений.
46	Даны два предложения. Найти самое короткое из слов первого предложения, которого нет во втором предложении.
47	Отредактировать заданное предложение, удаляя из него слова, которые встречаются в предложении заданное число раз.
48	Отредактировать заданное предложение, заменяя всякое вхождение слова вида *abc* на *b*, где *a,b*—подслова, *c* —обращение слова *a*.
49	Преобразовать строку таким образом, чтобы в ее начале были записаны слова, содержащие только цифры, потом слова, содержащие только буквы, а затем слова, которые содержат и буквы и цифры.
50	Заменить в строке символьные константы вида 'A' на соответствующие шестнадцатеричные (т.е. 'A' на 0x41).
51	Выполнить сортировку символов в строке. Порядок возрастания "весов" символов задать таблицей вида char ORD[] = "АаБбВвГгДдЕе1234567890"; Символы, не попавшие в таблицу, размещаются в конце отсортированной строки.

Контрольные вопросы

1. Запишите строку char `s[] = "ABCDEF";` через массив символов.
2. Каков размер строки `str` в следующем примере: `char str[] = "a short string";`?
3. Является ли выражение тождеством `strlen(s)==sizeof(s)` ?
4. Как можно интерпретировать указатель на `char`?
5. Являются ли описания char `s[];` и `char *s;` эквивалентными ?

Практикум № 9-10. МНОГОМЕРНЫЕ МАССИВЫ И УКАЗАТЕЛИ

Краткие теоретические сведения

В языке С определены только одномерные массивы (вектора), но поскольку элементом массива может быть массив, можно определить и многомерные массивы. Они формализуются списком константных-выражений следующих за идентификатором массива, причем каждое константное-выражение заключается в свои квадратные скобки. Константное-выражение в квадратных скобках определяет число элементов по данному измерению массива, так что объявление двухмерного массива содержит два константных-выражения, трехмерного – три и т.д. Пример:

```
int a[2][3]; /* представлено в виде матрицы
                a[0][0]  a[0][1] a[0][2]
                a[1][0]  a[1][1] a[1][2]      */
double b[10]; /* вектор из 10  элементов имеющих тип double */
int w[3][3] = { { 2, 3, 4 },
                { 3, 4, 8 },
                { 1, 0, 9 } };
```

В примере объявлен массив `w[3][3]`. Списки, выделенные в фигурные скобки, соответствуют строкам массива, в случае отсутствия скобок инициализация будет выполнена неправильно.

В языке С/С++ можно использовать сечения массива. Сечения формируются вследствие опускания одной или нескольких пар квадратных скобок. Пары квадратных скобок можно отбрасывать только справа налево и строго последовательно. Для `int s[2][3]` при обращении `s[0]` будет передаваться нулевая строка массива `s`.

Указатели на многомерные массивы – это массивы массивов, т.е. такие массивы, элементами которых являются массивы. Например при выполнении объявления двумерного массива `arr[4][3]` в памяти выделяется участок для хранения значения переменной arr, которая является указателем на массив из четырех указателей. Для этого массива из четырех указателей тоже выделяется память. Каждый из этих четырех указателей содержит адрес массива из трех элементов типа `int`, и, следовательно, в памяти компьютера выделяется четыре участка для хранения четырех массивов чисел типа `int`, каждый из которых состоит из трех элементов. Такое выделение памяти показано на схеме (рис. 6).

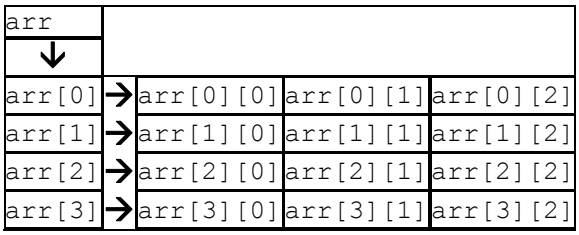


Рис. 6.

Для доступа к элементам двумерного массива должны быть использованы два индексных выражения в форме `arr[m][n]` или эквивалентных ей `*(*(arr+m)+n)`, `*(arr[m]+n)` и `(*(arr+m))[n]`. С точки зрения синтаксиса языка указатель `arr` и указатели `arr[0]`, `arr[1]`, `arr[2]`, `arr[3]` являются константами и их значения нельзя изменять во время выполнения программы.

При размещении элементов многомерных массивов они располагаются в памяти подряд по строкам. Такой порядок дает возможность обращаться к любому элементу многомерного массива, используя адрес его начального элемента и только одно индексное выражение:

```
float *ptr;
arr[2][3][4] ==ptr[3*2+4*3+4] ==ptr3[22];
```

Элементы массивов могут иметь любой тип, и, в частности, могут быть указателями на любой тип. Следующие объявления переменных

```
int a[]={10,11,12,13,14,};
int *p[]={a, a+1, a+2, a+2, a+3, a+4};
int **pp=p;
```

порождают программные объекты, представленные на схеме (рис. 7).

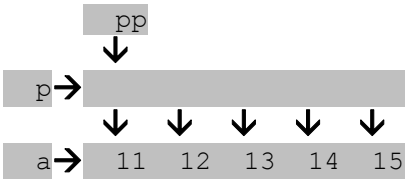


Рис. 7.

Если считать, что `pp=p`, то обращение `*++pp` это значение первого элемента массива a (т.е. значение 11), операция `++*pp` изменит содержимое указателя `p[0]`, таким образом, что он станет равным значению

адреса элемента `a[1]`.

Независимо от размерности массива, объем занимаей им памяти определяется достаточно прости, с помощью опреации `sizeof`:

```
Размер массива=sizeof имя_массива/sizeof(тип_данных);
```

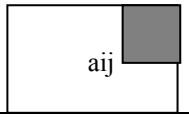
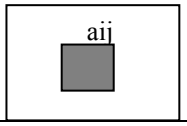
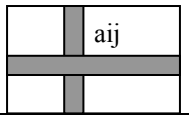
Задания для выполнения

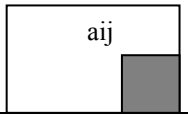
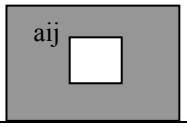
Таблица 19

Дана действительная матрица порядка nxm	
1	Поменять местами строку, содержащую элемент с наибольшим значением в матрице со строкой, содержащей элемент с наименьшим значением. Вывести на экран полученную матрицу. Для каждой строки с нулевым элементом на главной диагонали вывести ее номер и значение наибольшего из элементов этой строки.
2	Определить, является ли эта матрица симметричной (относительно главной диагонали). Вывести на экран соответствующее сообщение. Найти максимальный элемент среди стоящих на главной и побочной диагонали и поменять местами с элементом, стоящим на пересечении этих диагоналей.
3	Выведите номера столбцов, все элементы, которых четны. Для каждого столбца с отрицательным элементом на главной диагонали вывести его номер и сумму всех элементов этого столбца.
4	Среди строк заданной матрицы, содержащих только нечетные элементы, найти строку с максимальной суммой модулей элементов.
5	Среди столбцов заданной матрицы, содержащих только такие элементы, которые по модулю не больше n, найти столбец с минимальным произведением элементов.
6	Найти такие k, что k-я строка матрицы совпадает c k-м столбцом.
7	Матрица имеет седловую точку a_{ij}, если a_{ij} является минимальным в i-й строке и максимальным в j-м столбце. Найти все седловые точки заданной матрицы.
8	Получите номера строк, элементы каждой из которых образуют монотонную последовательность (монотонно убывающую или монотонно возрастающую).
9	Выведите номера столбцов, элементы каждой из которых образуют монотонную последовательность (монотонно убывающую или монотонно возрастающую).
10	Найти максимальный среди всех элементов тех строк заданной матрицы, которые упорядочены (либо по возрастанию, либо по убыванию).
11	Найти номер строки заданной матрицы, в которой находится самая длинная серия (последовательность одинаковых элементов).
12	Найти строку заданной матрицы, в которой длина максимальной серии (последовательности одинаковых элементов) минимальна.

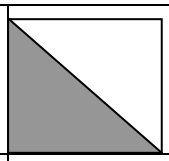
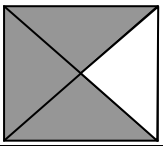
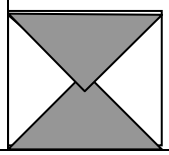
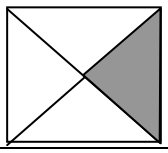
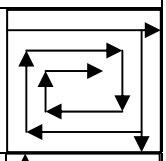
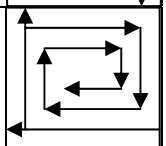
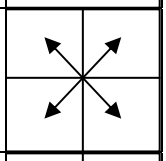
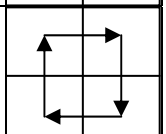
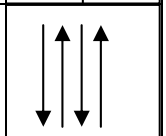
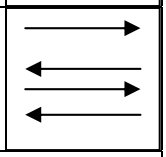
Продолжение таблицы 19

13	Найти максимальное из чисел, встречающихся в заданной матрице более одного раза.
14	Найти минимальное из чисел, встречающихся в заданной матрице ровно один раз.
15	Для заданной целочисленной матрицы найти максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.
16	Для заданной целочисленной матрицы найти минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали.
17	Подсчитать количество столбцов заданной матрицы, которые составлены из попарно различных чисел.
18	Подсчитать количество строк заданной матрицы, являющихся перестановкой чисел 1,2,...,20.
19	Пусть *m(A, i)* означает номер столбца матрицы *A*, в котором находится последний в строке минимум i-й строки. Проверить, что для заданной матрицы *A* выполняются неравенства $m(A,1) \le m(A,2) \le .. \le m(A,m)$.
20	Проверить, удовлетворяет ли заданная матрица *A* следующему условию: для всех $i > 1$ и для всех j> *1* верно неравенство $ajj \ge aj\text{-}1j + ajj\text{-}1$.
21	Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Соседями элемента a_{jj} в матрице назовем элементы a_{ki} с $i\text{-}1 \le k \le i+1,\ j\text{-}1 \le l \le j+1, (k,l) \ne (i,j)$. Подсчитать количество локальных минимумов заданной матрицы.
23	Найти минимум среди всех локальных максимумов заданной матрицы. Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Соседями элемента a_{jj} в матрице назовем элементы a_{ki} с $i\text{-}1 \le k \le i+1,\ j\text{-}1 \le l \le j+1, (k,l) \ne (i,j)$.
24	Определить, становится ли симметричной (относительно главной диагонали) заданная матрица после замены на число 0 каждого локального минимума. Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Соседями элемента a_{jj} в матрице назовем элементы a_{ki} с $i\text{-}1 \le k \le i+1,\ j\text{-}1 \le l \le j+1, (k,l) \ne (i,j)$.
25	Две строки матрицы назовем похожими, если совпадают множества чисел, встречающихся в этих строках. Найти количество строк в максимальном множестве попарно непохожих строк заданной матрицы.
26	Определить, является ли действительная матрица размера mxn ортонормированной, т.е. такой, в которой скалярное произведение каждой пары различных строк равно 0, а скалярное произведение каждой строки на себя равно 1. Вывести на экран соответствующее сообщение.
Получить новую матрицу	
27	Даны две действительные квадратные матрицы порядка *n*. Получить новую матрицу умножением элементов каждой строки первой матрицы на наибольшее из значений элементов соответствующей строки второй матрицы.

28	Даны две действительные квадратные матрицы порядка *n*. Получить новую матрицу, путем прибавления к элементам каждого столбца первой матрицы произведения элементов соответствующих строк второй матрицы.
29	Переставляя ее строки и столбцы, добиться, чтобы наибольший элемент (один из них) оказался в верхнем левом углу. Вывести на экран полученную матрицу.
30	В данной действительной квадратной матрице порядка *n* найти max по модулю элемент. Получить квадратную матрицу порядка *n-1* путем выбрасывания из исходной какой-либо строки и столбца, на пересечении которых расположен элемент с найденным значением.
31	Получить квадратную матрицу порядка *n+1* путем добавления к исходной какой-либо строки и столбца, на пересечении которых расположен элемент с найденным значением.
32	Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы матрицы, расположить их в соответствии с ростом характеристик.
33	Для двух заданных матриц одинакового размера проверить, можно ли получить вторую матрицу из первой применением (конечного числа раз) операций транспонирования относительно главной и побочной диагоналей.
Дана действительная матрица *[aij] i,j=1,...,n* . Получить действительную матрицу *[b ij] i,j=1,...,n* , элемент *b ij* которой равен	
34	сумме элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область). 
35	произведению элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область). 
36	максимуму из элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область). 
37	минимуму из элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область). 
38	произведению элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область). 

39	сумме элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область).				
40	максимуму из элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область).				
41	минимуму из элементов данной матрицы, расположенных в области, определяемой индексами *i,j* так, как показано на рис. (заштрихованная область).				
Получить квадратную матрицу порядка n					
42	n 0 ... 0 0 n-1 ... 0 ... ... 0 0 ... 1		51	1·2 0 ... 0 0 2·3 ... 0 ... ... 0 0 ... n·n+1	
43	1 2 3 ... 3 2 1 0 1 2 ... 2 1 0 ... ... 0 1 2 ... 2 1 0 1 2 3 ... 3 2 1		53	1 1 ... 1 1 0 1 ... 1 0 ... ... 0 1 ... 1 0 1 1 ... 1 1	
44	n n-1 ... 1 n-1 n-2 ... 0 ... ... 1 0 ... 0		55	1 2 ... n-1 n 2 3 ... n 0 ... ... n ... 0	
45	1 0 ... 0 1 1 1 ... 1 1 0... ... 1 1 ...0 1 1 ... ... 1 0 ... 0 1		57	n 0 ... 0 0 n-1 n ... 0 0 ... ... 2 3 ... n 0 1 2 ... n-1 n	
46	1 1 ... 1 1 2 2 ... 2 0 ... ... n 0 ... 0 0		59	1 0 ... 0 n 0 2 ... n-1 0 ... ... 0 2 ... n-1 0 1 0 ... 0 n	
Дана действительная матрица [a_{ij}] i,j=1,...,n . Найти сумму и мах значение среди элементов, расположенных в заштрихованной части матрицы					

Продолжение таблицы 19

47		48	
49		50	
Выполнить следующе задание			
51	Получить целочисленную квадратную матрицу порядка *n*, элементами которой являются числа 1,2,...,n^2 , расположенные по спирали рис.		
52	Дана действительная квадратная матрица порядка n. Вывести последовательность действительных чисел *b1,... ,bnxn*, получающуюся при чтении данной матрицы по спирали рис.		
53	Дана действительная квадратная матрица порядка *2*n. Получить новую, переставляя ее блоки размера nxn		
54	Дана действительная квадратная матрица порядка *2n*. Получить новую, переставляя ее блоки размера nxn		
55	Даны действительные числа *a1,...,a nxn* . Получить действительную квадратную матрицу порядка *n*, элементами которой являются числа *a1,...,a nxn*, расположенные по рис		
56	Даны действительные числа *a1,...,a nxn* . Получить действительную квадратную матрицу порядка *n*, элементами которой являются числа *a1,...,a nxn*, расположенные по рис		
Для заданной целочисленной матрицы *A* (*nxm)* построить матрицу *B*			
57	такого же размера по следующему правилу: в каждой строке матрицы *A* удалить одинаковые элементы, сдвинув элементы в строке, а в конец строк добавить нули.		
58	строки которой будут состоять из различных элементов соответствующей строки матрицы *A*. Количество столбцов построенной матрицы B равно наибольшему количеству различных элементов в строке, строки с меньшим числом элементов дополнить нулями.		

Окончание таблицы 19

59	такого же размера, элементы которой обладают следующим свойством: элемент *B[i,j]* равен минимальному из элементов матрицы *A*, расположенных ниже диагонали параллельной главной, пересекающей этот элемент, включая саму диагональ.
Задана символьная матрица *nxm*	
60	Заменить буквой 'а' все элементы, расположенные выше главной диагонали и не являющимися цифрами.
61	Заменить символом * все элементы, расположенные выше побочной диагонали и не являющиеся цифрами.
62	Определить номер последней строки, содержащей наименьшее число знаков "+" и "-".
63	Определить номер первой по порядку строки, содержащей наименьшее число букв 'ш', 'щ'.
64	Определить номер первой по порядку строки, содержащей наибольшее число цифр.
65	Получить последовательно все строки матрицы, исключая те, для которых есть равные среди строк с меньшими номерами
Выполнить задание	
66	На шахматной доске находятся король и несколько ферзей другого цвета. Проверить, находится ли король под угрозой и если да, кто ему угрожает. Положение фигур задано массивом *K(8, 8)*: 0 — клетка пуста, 1 — король, 2 — ферзь. Ферзь бьет по горизонтали, вертикали и диагоналям.
67	В трехмерном массиве *K(f,m,n)*, состоящем из нулей и единиц, хранится сеточное изображение некоторого трехмерного тела. Получить в двумерных массивах три проекции (тени) этого тела
68	Клеточное поле размером *mxn* есть результат игры в крестики-нолики на «бесконечном» поле. Проверить, не закончена ли игра выигрышем «крестиков»? Считается, что «крестики» выиграли, если на поле найдется по горизонтали, вертикали или диагонали цепочка, состоящая подряд из 5 крестиков.

Контрольные вопросы

1. Как определяется двумерный массив в C/C++?

2. Приведите примеры определений и инициализации многомерных массивов.

3. Сколько байт будет выделено под масиив `int a[2][2][3];`?

4. Пусть есть `mas[30][100]` обратитесь к элементу `mas[22][0]` двумя способами.

5. Продолжите запись `A+1==&A[` ...

Практикум №11. ФУНКЦИИ

Краткие теоретические сведения

Функция – это совокупность объявлений и операторов, обычно предназначенная для решения определенной задачи. Каждая функция должна иметь имя, которое используется для ее объявления, определения и вызова. В любой программе С должна быть функция с именем `main` (главная функция), именно с этой функции, в каком бы месте программы она не находилась, начинается выполнение программы.

С использованием функций связаны три понятия – определение функции (описание действий, выполняемых функцией), объявление функции (задание формы обращения к функции) и вызов функции.

- Определение функций

Определение функции имеет следующую форму:

```
[спецификатор-класса-памяти] [спецификатор-типа] имя-функции
        ([список-формальных-параметров])
{   тело-функции    }
```

Необязательный спецификатор-класса-памяти задает класс памяти функции, который может быть `static` или `extern`.

Спецификатор-типа функции задает тип возвращаемого значения. Если он не задан, то предполагается, что функция возвращает значение типа `int`.

Имя-функции – либо `main` для основной функции, либо произвольный идентификатор, не совпадающий со служебными словами и именами других объектов программы.

Список-формальных-параметров – это последовательность объявлений формальных параметров вида <обозначение_типа> <имя_параметра>, разделенная запятыми. Формальные параметры – это переменные, используемые внутри тела функции и получающие значение при вызове функции путем копирования в них значений соответствующих фактических параметров. Список-формальных-параметров может заканчиваться запятой (,) или запятой с многоточием (,...), это означает, что число аргументов функции переменно.

Если функция не использует параметров, то наличие круглых скобок обязательно, а вместо списка параметров рекомендуется указать слово `void`.

Тело функции – это часть определения функции, заключенная в фигурные скобки `{ }`, содержащий операторы, определяющие действие функции.

Для передачи результата из функции в вызывающую функцию используется оператор return. Он может использоваться в двух формах: return; – завершает функцию не возвращающую никакого значения (т. е. перед именем функции указан тип void) и return <выражение>; – возвращает значение выражения, выражение должно иметь тип, указанный перед именем функции. Если программист не пишет оператор return явно, то компилятор автоматически дописывает return в конец тела функции перед закрывающей фигурной скобкой.

Функция не может возвращать массив или функцию, но может возвращать указатель на любой тип, в том числе и на массив, и на функцию. Тип возвращаемого значения, задаваемый в определении функции, должен соответствовать типу в объявлении этой функции.

Пример:

```
int rus (unsigned char r)
       {  if (r>='A' && r<=' ')
               return 1;
          else
               return 0; }
```

Все переменные, объявленные в теле функции без указания класса памяти, имеют класс памяти auto, т.е. они являются локальными. При вызове функции локальным переменным отводится память в стеке и производится их инициализация. Управление передается первому оператору тела функции и начинается выполнение функции, которое продолжается до тех пор, пока не встретится оператор return или последний оператор тела функции. Управление при этом возвращается в точку, следующую за точкой вызова, а локальные переменные становятся недоступными. При новом вызове функции для локальных переменных память распределяется вновь, и поэтому старые значения локальных переменных теряются.

Параметры функции передаются по значению и могут рассматриваться как локальные переменные, для которых выделяется память при вызове функции и производится инициализация значениями фактических параметров. При выходе из функции значения этих переменных теряются. Поскольку передача параметров происходит по значению, в теле функции нельзя изменить значения переменных в вызывающей функции, являющихся фактическими параметрами. Однако, если в качестве параметра передать указатель на некоторую переменную, то используя операцию разадресации можно изменить значение этой переменной. Пример:

```
void change (int *x, int *y)
    {  int k=*x;
          *x=*y;
          *y=k;   }
```

Определения используемых функций могут следовать за определением функции main, перед ним, или находится в другом файле. Для того, чтобы компилятор мог осуществить проверку соответствия типов передаваемых фактических параметров типам формальных параметров до вызова функции нужно поместить объявление (прототип) функции.

- Объявление функций

Если требуется вызвать функцию до ее определения в рассматриваемом файле, или определение функции находится в другом исходном файле, то вызов функции следует предварять объявлением этой функции. Объявление (прототип) функции имеет следующий формат:

```
[спецификатор-класса-памяти] [спецификатор-типа] имя-функции ([список-формальных-параметров]) [,список-имен-функций];
```

В отличие от определения функции, в прототипе за заголовком сразу же следует точка с запятой, а тело функции отсутствует. Если несколько разных функций возвращают значения одинакового типа и имеют одинаковые списки формальных параметров, то эти функции можно объявить в одном прототипе, указав имя одной из функций в качестве имени-функции. Правила использования остальных элементов формата такие же, как при определении функции. Имена формальных параметров при объявлении функции можно не указывать, а если они указаны, то их область действия распространяется только до конца объявления.

Тип возвращаемого значения при объявлении должен соответствовать типу возвращаемого значения в определении функции. Если прототип функции не задан, а встретился вызов функции, то строится неявный прототип из анализа формы вызова функции.

Пример прототипа:

```
int rus (unsigned char r);
```

или `rus (unsigned char);`

- Вызов функций

Вызов функции имеет следующий формат:

```
адресное-выражение ([список-выражений])
```

Поскольку синтаксически имя функции является адресом начала

тела функции, в качестве обращения к функции может быть использовано адресное-выражение (в том числе и имя функции или разадресация указателя на функцию), имеющее значение адреса функции.

Список-выражений представляет собой список фактических параметров, передаваемых в функцию. Этот список может быть и пустым, но наличие круглых скобок обязательно. Между формальными и фактическими параметрами должно быть соответствие по типам.

Адресное выражение, стоящее перед скобками определяет адрес вызываемой функции. Это значит, что функция может быть вызвана через указатель на функцию. Пример:

```
int (*fun)(int x, int *y);
```

Здесь объявлена переменная `fun` как указатель на функцию с двумя параметрами: типа `int` и указателем на `int`. Сама функция должна возвращать значение типа `int`. Круглые скобки, содержащие имя указателя `fun` и признак указателя `*`, обязательны, иначе запись

```
int *fun (intx,int *y);
```

будет интерпретироваться как объявление функции fun возвращающей указатель на int.

Вызов функции возможен только после инициализации значения указателя `fun` и имеет вид:

```
(*fun)(i,&j);
```

Любая функция в программе на языке С может быть вызвана рекурсивно, т.е. она может вызывать саму себя. Компилятор допускает любое число рекурсивных вызовов. Пример рекурсии – это математическое определение факториала n!:

```
long fakt(int n)
   { return ( (n==1) ? 1 : n*fakt(n-1)  );  }
```

- Вызов функции с переменным числом параметров

При вызове функции с переменным числом параметров задается любое требуемое число аргументов. В объявлении и определении такой функции переменное число аргументов задается многоточием в конце списка формальных параметров или списка типов аргументов.

Примерами функций с переменным числом параметров являются функции из библиотеки функций (`printf`,`scanf` и т.п.).

- Параметры функции по умолчанию

Часто в самом общем случае функции требуется больше параметров, чем в самом простом и более употребительном случае. Если

функция описывается :

```
extern char* func(long, int =0);
```

то инициализатор второго параметра является параметром по умолчанию. То есть, если в вызове дан только один параметр, в качестве второго используется параметр по умолчанию. Например:

```
cout << func(31) << func(32,3);
```

интерпретируется как

```
cout << func(31,0) << func(32,3);
```

Параметр по умолчанию проходит проверку типа во время описания функции и вычисляется во время ее вызова. Задавать параметр по умолчанию возможно только для последних параметров.

- Перегрузка имен функций

В C++ возможно определение нескольких функций с одинаковым именем, но с разными типами формальных параметров и результата. При этом транслятор выбирает соответствующую функцию по типу фактических параметров. Переопределяемую функцию необходимо объявить с ключевым словом `overload`, а можно и без нее. Например:

```
overload print;
void print(int);
void print(char*);
```

- Передача параметров функции main

Функция `main` может быть определена с параметрами, которые передаются из внешнего окружения, например, из командной строки. Во внешнем окружении все данные представляются в виде строк символов. Для передачи этих строк в функцию `main` используются два параметра, первый параметр служит для передачи числа передаваемых строк, второй для передачи самих строк. Общепринятые (но не обязательные) имена этих параметров `argc` и `argv`. Параметр `argc` имеет тип `int`, его значение формируется из анализа командной строки и равно количеству слов в командной строке, включая и имя вызываемой программы (под словом понимается любой текст не содержащий символа пробел). Параметр argv это массив указателей на строки, каждая из которых содержит одно слово из командной строки. Если слово должно содержать символ пробел, то при записи его в командную строку оно должно быть заключено в кавычки.

Функция `main` может иметь и третий параметр, который принято

называть `argp`, и который служит для передачи в функцию `main` параметров операционной системы (среды) в которой выполняется С-программа. Заголовок функции `main` имеет вид:

```
int main (int argc, char *argv[], char *argp[])
```

Если командная строка имеет вид:

```
A:\>cprog working 'C program' 1
```

то аргументы `argc`, `argv`, `argp` представляются в памяти следующим образом:

```
     argc  [ 4  ]
     argv  [    ]--> [    ]-->   [A:\cprog.exe\0]
                     [    ]-->   [working\0]
                     [    ]-->   [C program\0]
                     [    ]-->   [1\0]
                     [NULL]
     argp  [    ]--> [    ]-->   [path=A:\;C:\\0]
                     [    ]-->   [lib=D:\LIB\0]
                     [    ]-->   [include=D:\INCLUDE\0]
                     [    ]-->   [conspec=C:\COMMAND.COM\]
                     [NULL]
```

- Указатели как формальные параметры и результат функций

Если в качестве параметра функции используется обозначение массива или строки, то в функцию передается адрес первого элемента массива или строки соответсвенно. Пример:

```
//вычисление суммы элементов массива вариант 1
int sum (int n, int a[] )
   {  int i,int s=0;
      for( i=0; i<n; i++ )
      s+=a[i]
      return s;
   }
void main()
   {  int a[]={ 3, 5, 7, 9, 11, 13, 15 };
      int s = sum( 7, a );
      cout<<s;     }
//вариант 2
int sum (int n, int *a)
   {  for(int i=0, s=0; i<n; s+=*(a+i),i++ );
      return s;    }
void main()
  {   int a[]={ 3, 5, 7, 9, 11, 13, 15 };
      int s = sum( 7, a );
      cout<<s;   }
```

Функция в качестве результата может возвращать указатель. В этом случае она обычно «выбирает» некоторую переменную из имеющихся или же «создает» ее:

```
int   *min(int A[], int n)
// функция возвращает указатель на минимальный элемент массива
{
int   *pmin, i;   // Рабочий указатель, содержащий результат
for (i=1, pmin=A; i<n; i++)
      if (A[i] < *pmin)   pmin = &A[i];
return(pmin);
}
```

Указатель – результат функции может ссылаться не только на отдельную переменную, но и на массив.

- Указатель на функцию

Указателем на функцию называется переменная, которая содержит адрес некоторой функции. Соответственно, косвенное обращение по этому указателю представляет собой вызов функции. Определение указателя на функцию имеет вид:

```
int     (*pf)();               // без контроля параметров вызова
int     (*pf)(int, char*);     // с контролем по прототипу
```

Выражение вида `&имя_функции` имеет смысл – начальный адрес функции или указатель на функцию. По аналогии с именем массива использование имени функции без скобок интерпретируется как указатель на эту функцию. Указатель может быть инициализирован и при определении. Возможны следующие способы назначения указателей:

```
int   INC(int a) { return a+1; }
extern      int  DEC(int);
int   (*pf)(int);
pf = &INC;
pf = INC;                      // присваивание указателя
int   (*pp)(int) = &DEC;       // инициализация указателя
```

Указатели на функции могут входить в более сложные структуры данных. Указатель на функцию может быть передан в качестве формального параметра.

Задания для выполнения

Таблица 20

Вариант задания реализовать в виде функции, использующей для работы со строкой и массивами (матрицами) только указатели и операции вида *p++, p++ и т.д. Если функция возвращает строку или ее фрагмент, то это также необходимо сделать через указатель.	
1	Функция находит в строке самое первое (по алфавиту) слово. С ее помощью реализовать размещение слов в выходной строке в алфавитном порядке.

Продолжение таблицы 20

2	Функция создает копию строки и "переворачивает" в строке все слова. Пример: "Жили были дед и баба" - "илиЖ илиб дед и абаб".
3	Из двумерного массива в одномерный записали сначала строки в произвольном порядке, затем столбцы в произвольном порядке. Написать функцию восстанавливающую исходный массив по одномерному, если известна размерность двумерного массива и элементы в нем не повторяются
4	Функция находит в строке симметричный фрагмент вида "abcdcba" длиной 7 и более символов (не содержащий пробелов) и возвращает указатель на его начало и длину. С использованием функции “вычеркнуть” все симметричные фрагменты из строки
5	Функция находит в строке пары инвертированных фрагментов (например "123apr" и "rpa321") и возвращает указатель на первый. С помощью функции найти все пары.
6	Функция находит в строке заданную подстроку и возвращает указатель на нее. С использованием функции найти все вхождения подстроки в строке.
7	Функция находит в строке десятичные константы и заменяет их на шестнадцатеричные с тем же значением, например "aaaaa258xxx" на "aaaaa0x102xxx".
8	Написать функцию для обмена строк двумерного массива с ее помощью отсортировать массив по элементам третьего столбца.
9	Написать функцию для суммирования матриц. С ее помощью сложить исходную матрицу и транспонированную (т. е. полученную поворотом исходной на 90 °.
10	Написать функцию для удаления строки из двумерного массива. Оставшиеся строки должны быть расположены плотно, недостающие элементы заменяются 0. С помощью разработанных функций исключить из массива строки с номерами от *A* до *B*.
11	Написать функцию обмена столбца и строки двухмерного массива. С ее помощью поменять местами те строки и столбцы, первые элементы которых совпадают.
12	Написать функцию транспонирования квадратной матрицы (т.е. поворота исходной матрицы на 90 °). С ее помощью определить является ли заданная матрица симметрической. (Матрица называется симметрической, если транспонированная матрица равна исходной).
13	Написать функцию для вычисления суммы элементов квадратной матрицы, которые расположены ниже главной диагонали. С ее помощью найти максимальное значение такой суммы в *n* матрицах.
14	Написать функцию, проверяющую есть ли отрицательные элементы в указанной строке двумерного массива. Удалить из массива все строки с отрицательными элементами, удаленная строка заполняется 0 и переносится в конец массива.

Продолжение таблицы 20

15	Написать функцию, для поиска максимального элемента в указанной строке двумерного массива. Сдвинуть в двумерном массиве все строки циклически вправо на количество элементов равное максимальному элементу в этой строке.
16	Задан двумерный массив. Найти сумму элементов первого столбца без одного последнего элемента, сумму элементов второго столбца без двух последних, сумму элементов третьего столбца без трех последних и т. д. Последний столбец не обрабатывается. Среди найденных сумм найти максимальную.
17	Задан двумерный массив *N x N*. Разрешается произвольно переставлять элементы внутри любого столбца. Проверить, можно ли выполнив конечное количество перестановок в столбцах, расположить на побочной диагональ элементы так, чтобы он возрастали.
18	Задан двумерный массив *N x M*. Найти в нем подмассив 3 x 3, сумма элементов которого максимальна. *N* и *M* могут быть не кратны трем.
19	Задан двумерный массив *N x N*. Последовательно рассматриваются квадратные подмассивы, правый верхний элемент которых лежит на побочной диагонали. В каждом таком подмассиве находится максимальный элемент. Путем перестановок строк и столбцов (целиком) элемент надо переместить в правый верхний угол подмассива. Проверить получилась ли на побочной диагонали убывающая последовательность элементов.
Решить указанную в варианте задачу, используя функции с переменным числом параметров	
20	Написать функцию sum с переменным числом параметров, которая находит сумму чисел типа int по формуле: *S=a1*a2+a2*a3+a3*a4+. . . .* Написать вызывающую функцию main, которая обращается к функции sum не менее трех раз с количеством параметров 5, 10, 12.
21	Написать функцию min с переменным числом параметров, которая находит минимальное из чисел типа int. Написать вызывающую функцию main, которая обращается к функции min не менее трех раз с количеством параметров 5, 10, 12.
22	Написать функцию max с переменным числом параметров, которая находит минимальное из чисел типа int или из чисел типа double, тип параметров определяется с помощью первого параметра функции. Написать вызывающую функцию main, которая обращается к функции min не менее трех раз с количеством параметров 5, 10, 12.
23	Написать функцию days с переменным числом параметров, которая находит количество дней, прошедших между двумя датами (параметрами функции являются даты в формате «дд.мм.гг». Написать вызывающую функцию main, которая обращается к функции days не менее трех раз с количеством параметров 3, 5, 8.

Окончание таблицы 20

Выполнить задания с использованием функций двумя способами: с параметрами, без параметров	
24	Дан массив целых чисел, содержащий *n* элементов. Записать в этот же массив сначала все положительные числа, а затем все отрицательные и нули, сохраняя порядок их следования.
25	Дан массив целых чисел, содержащий *n* элементов Из массива выбрать без повторений все элементы, встречающиеся более одного раза.
26	Получить массив *C(K)*, упорядоченный по возрастанию, путем слияния массивов *A(N)* и *B(M)*, упорядоченных по возрастанию *(K=N+M).*
27	Дан массив целых чисел, содержащий *n* элементов Элементы массива циклически сдвинуть на *K* позиций вправо.
28	По заданной квадратной матрице размером *nxn* построить вектор длиной *2n-1*, элементы которого – максимумы элементов, диагоналей, параллельных главной диагонали.
Написатьь функцию для выполнения следующих заданий через перебор с возвратом или рекурсию	
29	Дано натуральное *n*.Получить все магические квадраты размерности 3x3, элементы в которых <=*n*
30	Получить все перестановки элементов 1, .., 6
31	Получить все расстановки 8 ладей на шахматной доске, при которых ни одна ладья не угрожает другой
32	Задача о 8 слонах: на шахматной доске расставить 8 слонов так, чтобы каждое поле находилось под ударом одного из них
33	На одной из клеток шахматной доски стоит конь. Требуется выполнить обход конем шахматной доски. Ни одну из клеток конь не может проходить дважды, но каждой клетке он обязан побывать. Выдать сообщение, если обхода не существует
34	Найти расстановку 5 ферзей на шахматной доске, при которой каждое поле будет находиться под ударом одного из них
35	Найти такую расстановку 12 коней на шахматной доске, при которой каждое поле будет находиться под ударом одного из них

Контрольные вопросы

1. В чем заключаются различия между библиотечными функциями и функциями, которые вы пишете сами?

2. В чем разница между объявлением и определением функции, приведите примеры?

3. Как передать и получить значение функции?

4. Как выполнить передачу параметра в функцию по адресу?

5. Как передать массив в функцию?

Практикум №12. СОРТИРОВКА И ПОИСК

Краткие теоретические сведения

Алгоритмы сортировки можно классифицировать по нескольким признакам. По размещению элементов сортировки бывают внутренними – в памяти и внешними – в файле данных. По виду структуры данных, содержащей сортируемые элементы можно выделить сортировки массивов, массивов указателей, списков и других структур данных. По способу выбора элементов сортировки делятся на сортировки подсчетом, вставками, выбором, слиянием, разделением, обменные сортировки.

- Обменныая или пузырьковая сортировка

Наиболее простой метод систематического обмена соседних элементов. При просмотре всего массива слева на право соседние элементы, в случае расположения не в порядке возрастания, переставляются. Просмотр повторяется до тех пор, пока перестановок больше не будет. Пример:

```
B=<20,-5,10,8,7>,   исходный массив;
B1=<-5,10,8,7,20>,  первый просмотр;
B2=<-5,8,7,10,20>,  второй просмотр;
B3=<-5,7,8,10,20>,  третий просмотр.
```

Пузырьковая сортировка не требует дополнительной памяти.

- Быстрая сортировка разделением

Быстрая сортировка состоит в том, что массив B реорганизуется в B', B'', где B' – неотсортированный подмассив B с элементами, не большими K (медианы), а B'' – неотсортированный подмассив B с элементами, большими K. В B', B'' элемент K расположен на месте, на котором он должен быть в результирующем отсортированном массиве. Далее к подмассивам B' и B'' снова применяется упорядочивание быстрой сортировкой. Приведем в качестве примера сортировку массива, отделяя упорядоченные элементы косой чертой, а элементы Ki знаками <>:

```
9, 7, 18, 3, 52, 4, 6, 8, 5, 13, 42, 30, 35, 26
7, 3, 4, 6, 8, 5/ <9>/ 18, 52, 13, 42, 30, 35, 26
3, 4, 6, 5/<7>/ 8/ 9/ 13/ <18>/ 52, 42, 30, 35, 26
<3>/ 4, 6, 5/ 7/ 8/ 9/ 13/ 18/ 42, 30, 35, 26/ <52>
3/ <4>/ 6, 5/ 7/ 8/ 9/ 13/ 18/ 30, 35, 26/ <42>/ 52
3/ 4/ 5/ <6>/ 7/ 8/ 9/ 13/ 18/ 26/ <30>/ 35/ 42/ 52
```

Время работы методом быстрой сортировки зависит от упорядоченности массива. Оно будет минимальным, если на каждом шаге

разбиения получаются подмассивы *B′* и *B″* приблизительно равной длины, и тогда требуется около *n*log2(n)* шагов. Если массив близок к упорядоченному, то требуется около *(n*n)/2* шагов.

- Сортирвка подсчетом

Упорядоченный массив *B′* получается из *B* путем сравнения всех пар элементов массива для каждого из них подсчитывается количество элементов, меньших его. Это дает новое местоположение этого элемента в выходном массиве:

```
B=<20,-5,10,8,7>,
S=<4,  0, 3,2,1>,
B′=<-5,7,8,10,20>.
```

- Сортировка выбором

Упорядоченный массив *B′* получается из *B* многократным применением выборки из *B* минимального элемента, удалением этого элемента из *B* и добавлением его в конец *B*', который первоначально должен быть пустым. Пример:

```
B=<20,10,8,-5,7>, B'=<>
B=<20,10,8,7>,    B'=<-5>
B=<20,10,8>,      B'=<-5,7>
B=<20,10>,        B'=<-5,7,8>
B=<20>,           B'=<-5,7,8,10>
B=<>,             B'=<-5,7,8,10,20>.
```

Сортировка квадратичной выборкой. Исходный массив *B* из *n* элементов делится на *m* подмассивов *B1,B2,...,Bm*, где *m* равно квадратному корню из *n*, и в каждом B находится минимальный элемент *G*. Наименьший элемент всего массива *B* определяется как минимальный элемент *Gj*, и выбранный элемент *Gj* заменяется новым наименьшим из массива *Bj*. Количество действий, требуемое для сортировки квадратичной выборкой *Q=n*n*, но требуется дополнительная память для хранения *G*.

- Сортировка вставкой

Упорядоченный массив *B′* получается из *B* следующим образом: сначала он состоит из единственного элемента *K1*; далее для *i=2,...,n* выполняется вставка узла *Ki* в *B′* так, что *B′* остается упорядоченным массивом длины *i*. Например, для начального списка `B=<20,-5,10,8,7>`:

```
B=<20,-5,10,8,7>  B'=<>
B=<-5,10,8,7>     B'=<20>
B=<10,8,7>        B'=<-5,20>
B=<8,7>           B'=<-5,10,20>
B=<7>             B'=<-5,8,10,20>
B=<>              B'=<-5,7,8,10,20>
```

- Сортировка слиянием

Для получения упорядоченного массива *B'* последовательность значений *B* разделяют на *n* массивов *B1, B2,...,Bn*, длина каждого из которых 1. Затем осуществляется функция прохода, при которой *m>=2* упорядоченных массивов *B1,B2,...,Bm* заменяется на *m/2* (или *(m+1)/2)* упорядоченных массивов, *B(2i-1*)-ого и *B(2i)*-ого *(2i<=m)* и добавлением *Bm* при нечетном *m*. Проход повторяется до тех пор пока не получится одна последовательность длины *n.*

Приведем пример сортировки массива путем использования слияния, отделяя последовательности косой чертой, а элементы запятой:

```
9 / 7 / 18 / 3 / 52 / 4 / 6 / 8 / 5 / 13 / 42 / 30 / 35 / 26;
7,9 / 3,18 / 4 / 52 / 6 / 8 / 54 / 13 / 30 / 42 / 26 / 35;
3,7,9,18 /  4,6,8,52 /  5,13,30,42 /  26,35;
3,4,6,7,8,9,18,52 /   5,13,26,30,35,42;
3,4,5,6,7,8,9,13,18,26,30,35,42,52.
```

Количество действий равно *Q=n*log2(n)*, так как за один проход выполняется *n* сравнений, а всего необходимо осуществить *log2(n)* проходов. Для сортировки слиянием удобно использовать рекурсию.

- Распределяющая сортировка

Предположим, что элементы массива *B* есть *T*-разрядные положительные десятичные числа *D(j,n) – j*-я справа цифра в десятичном числе n>=0, т.е. *D(j,n)=floor(n/m)%10*, где *m=10^(j-1)*. Пусть *B0,B1,...,B9* – вспомогательные массивы (карманы), вначале пустые. Для реализации распределяющей сортировки выполняется процедура, состоящая из двух процессов, называемых распределение и сборка для *j=1,2,...,T*.

РАСПРЕДЕЛЕНИЕ заключается в том, что элемент *Ki (i=1,n)* из *B* добавляется как последний в массив *Bm*, где *m=D(j,Ki)*, и таким образом получаем десять массивов, в каждом из которых *j*-тые разряды чисел одинаковы и равны *m*. СБОРКА объединяет массивы *B0,B1,...,B9* в этом же порядке, образуя один *B*.

Рассмотрим реализацию распределяющей сортировки при *T=2*:

```
B=<09,07,18,03,52,04,06,08,05,13,42,30,35,26> .
                  РАСПРЕДЕЛЕНИЕ-1:
B0=<30>,  B1=<>,  B2=<52,42>, B3=<03,13>, B4=<04>,
B5=<05,35>,  B6=<06,26>,B7=<07>, B8=<18,08>, B9=<09>.
                  СБОРКА-1:
B=<30,52,42,03,13,04,05,35,06,26,07,18,08,09>
                  РАСПРЕДЕЛЕНИЕ-2:
B0=<03,04,05,06,07,08,09>, B1=<13,18>, B2=<26>,
```

```
B3=<30,35>, B4=<42>, B5=<52>, B6=<>,B7=<>,B8=<>, B9=<>.
                СБОРКА-2:
B=<03,04,05,06,07,08,09,13,18,26,30,35,42,52>.
```

Количество действий, необходимых для сортировки *n T*-цифровых чисел, определяется как *Q(n*T)*. Недостатком этого метода является необходимость использования дополнительной памяти.

Задания для выполнения

Таблица 21

Алгоритм сортировки реализовать в виде функции, возвращающей характеристику трудоемкости алгоритма (количество сравнений, сдвигов). Предусмотреть заполение массива случайными значениями	
1	Сортировка вставками, место помещения очередного элемента в отсортированную часть производить с помощью двоичного поиска. Двоичный поиск оформить в виде отдельной функции.
2	Сортировка Шелла. Частичную сортировку с заданным шагом, начиная с заданного элемента оформить в виде функции. Алгоритм частичной сортировки - вставка погружением.
3	Сортировка разделением. Способ разделения: вычислить среднее арифметическое всех элементов массива и относительно этого значения разбить массив на две части (с использованием вспомогательных массивов).
4	Шейкер-сортировка. Процесс движения в прямом и обратном направлении реализовать в виде одного цикла, используя параметр - направление движения (+1/-1) и меняя местами нижнюю и верхнюю границы просмотра.
5	"Быстрая" сортировка с итерационным циклом вычисления медианы. Для заданного интервала массива, в котором производится разделение, ищется медиана обычным способом. Затем выбирается та часть интервала между границей и медианой, в которой находится середина исходного интервала, и процесс повторяется.
6	Сортировка циклическим слиянием с использованием одного выходного и двух входных массивов. Для упрощения алгоритма и разграничения сливаемых групп в последовательности в качестве разделителя добавить “очень большое значение” (MAXINT).
7	Сортировка разделением. Способ разделения: интервал между минимальным и максимальным значениями элементов массива разбить пополам и относительно этого значения разбить массив на две части (с использованием вспомогательных массивов).
8	Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы сдвигаются на один влево. Сам элемент заносится на освободившуюся последнюю позицию.

Окончание таблицы 21

9	Сортировка подсчетом. Выходной массив заполняется значениями “-1”. Затем для каждого элемента определяется его место в выходном массиве путем подсчета количества элементов строго меньших данного. Естественно, что все одинаковые элементы попадают на одну позицию, за которой следует ряд значений “-1”. После чего оставшиеся в выходном массиве позиции со значением “-1” заполняются копией предыдущего значения.
10	Простое однократное слияние. Разделить массив на n частей и отсортировать их произвольным методом. Отсортированный массив получить путем однократного слияния упорядоченных частей. Для извлечения очередных элементов из упорядоченных массивов использовать массив из n индексов (по одному на каждый массив).
11	Сортировка вставками. Берется очередной элемент и извлекается из массива. Затем от начала массива ищется первый элемент, больший данного. Все элементы, от найденного до очередного сдвигаются на один вправо и на освободившееся место помещается очередной элемент. (Поиск места включения от начала упорядоченной части).
12	Сортировка выбором. Выбирается минимальный элемент в массиве, переносится в выходной массив на очередную позицию и заменяется во входном на “очень большое значение” (MAXINT).
13	Сортировка Шелла. Частичную сортировку с заданным шагом, начиная с заданного элемента оформить в виде функции. Алгоритм частичной сортировки - обменная (методом “пузырька”).
14	Сортировка выбором. Выбирается минимальный элемент в массиве, переносится в выходной массив на очередную позицию. Во входном массиве все элементы от следующего за текущим до конца сдвигаются на один влево.
15	Сортировка “хитрая”. Из массива путем однократного просмотра выбирается последовательность элементов, находящихся в порядке возрастания, переносятся в выходной массив и заменяются во входном на “-1”. Затем оставшиеся элементы включаются в полученную упорядоченную последовательность методом погружения.

Контрольные вопросы

1. Какие существуют классификации алгоритмов сортировки?
2. Подсчитайте количество действий для сортировки вставками.
3. Назовите методы сортировки, которые требуют дополнительной памяти.
4. От чего может зависеть скорость методов сортировки?
5. Приведите другой вариант сортировки слиянием, отличный от описанного.

Практикум № 13-14. ДИНАМИЧЕСКИЕ ОБЪЕКТЫ

Краткие теоретические сведения

При традиционном определении массива:

```
тип имя_массива [количество_элементов];
```

общее количество памяти, выделяемой под массив, задается определением и равно `количество_элементов * sizeof(тип)`.

Но иногда бывает нужно, чтобы память под массив выделялась для решения конкретной задачи, причем ее размеры заранее не известны и не могут быть фиксированы. Формирование массивов с переменными размерами можно организовать с помощью указателей и средств динамического распределения памяти двумя способами: с использованием библиотечных функций (С); с использованием операций `new` и `delete` (С++).

- Формирование динамических массивов с использованием библиотечных функций

Для выделения и освобождения динамической памяти используются функции приведенные в табл. 22

Таблица 22

Функция	Прототип и краткое описание
`malloc`	`void * malloc(unsigned s)` Возвращает указатель на начало области динамической памяти длиной в s байт, при неудачном завершении возвращает NULL
`calloc`	`void * calloc(unsigned n, unsigned m)` Возвращает указатель на начало области динамической памяти для размещения n элементов длиной по m байт каждый, при неудачном завершении возвращает NULL
`realloc`	`void * realloc(void * p, unsigned s)` Изменяет размер блока ранее выделенной динамической памяти до размера s байт, p- адрес начала изменяемого блока, при неудачном завершении возвращает NULL
`free`	`void *free(void p)` Освобождает ранее выделенный участок динамической памяти, p – адрес первого байта

Для выделения памяти используется функция `malloc`, параметром которой является размер выделяемого участка памяти равный `n*sizeof(int)`. Так как функция `malloc` возвращает нетипизированный указатель `void*`, то необходимо выполнить преобразование получен-

ного нетипизированного указателя в необходимый тип указатель. Освободить выделенную память можно функцией `free()`:

```
//Функция для формирования одномерного динамического массива
#include <alloc.h>
int * make_arr(int n)
   {
      int *arr;
      arr=(int*)malloc(n*sizeof(int));
      for(int i=0;i<n;i++)
      arr[i]=rand();
      return arr;
    }
```

- Формирование динамических массивов с использованием операций new и delete

Для динамического распределения памяти используются операции new и delete. Формат:

```
new имя_типа
```

или

```
new имя_типа инициализатор
```

позволяет выделить и сделать доступным свободный участок памяти, размеры которого соответствуют типу данных, определяемому именем типа. В выделенный участок заносится значение, определяемое инициализатором, который не является обязательным параметром. В случае успешного выделения памяти операция возвращает адрес начала выделенного участка памяти, если участок не может быть выделен, то возвращается `NULL`. Примеры:

```
int *i;
i=new int(10);
...
float *f;
f=new float;
int *mas=new[5];
```

Операция `delete` указатель освобождает участок памяти ранее выделенный операцией new.

При формировании матрицы сначала выделяется памяти для массива указателей на одномерные массивы, а затем в цикле с параметром выделяется память под *n* одномерных массивов (рис. 8):

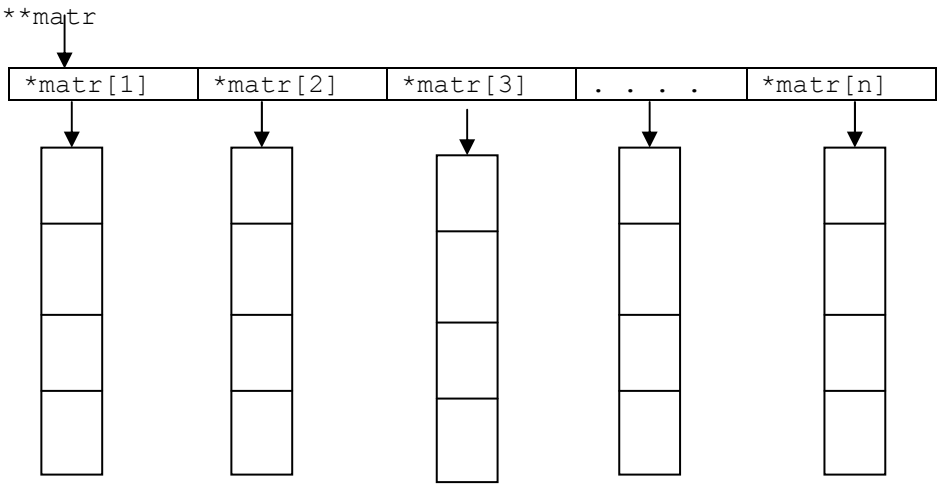


Рис. 8

Пример:

```
// Функция для формирования двумерного динамического массива
int ** make_matr(int n)
{
      int **matr;
      int i,j;
      matr=new int*[n];
      for (i=0;i<n;i++)
            {
            matr[i]=new int[n];
            for (j=0;j<n;j++)
            matr[i][j]=random(10);
            }
      return matr;
}
```

Чтобы освободить память необходимо выполнить цикл для освобождения одномерных массивов:

```
for(int i=0;i<n;i++)
delete matr[i];
//После этого освобождаем память на которую указывает указатель matr
delete [] matr;
```

- Массивы указателей

Массив указателей является самой простой и одновременно самой распространенной динамической структурой данных. В языке С одно из его определений имеет вид:

```
double      *p[20];
```

В соответствии с принципом контекстного определения типа данных переменную `p` следует понимать как массив, каждым элемен-

том которого является указатель на переменную типа `double`. Исходя из принятой в С концепции указателя, эту структуру можно рассматривать как массив указателей на отдельные переменные типа `double`, так и на массивы этих переменных:

```
char  *pc[] = { "aaa", "bbb", "ccc", NULL};
```

Массивы указателей как и все остальные структуры данных, содержащие указатели, допускают различные способы формирования, которые отличаются как способом создания самих элементов, так и способом установления связей между ними. Например:

```
double a1,a2,a3, *pd[] = { &a1, &a2, &a3, NULL};
...
double d[19], *pd[20];
for (i=0; i<19; i++)    pd[i] = &d[i];
pd[i] = NULL;
...
double      *p, *pd[20];
for (i=0; i<19; i++)
      {p = new double; *p = i;  pd[i] = p; }
pd[i] = NULL;
...
double      **pp, *p;
pp = new double *[20];
for (i=0; i<19; i++)
      {
      p = new double; *p = i; pp[i] = p;
      }
pp[i] = NULL;
```

Задания для выполнения

Решите задачи из вашего варианта тем «Массивы», «Строки» («Матрицы») с использованием динамических массивов и указателей. Размерности вводятся с клавиатуры. Применение в программе функций пользователя обязательно.

Контрольные вопросы

1. В чем разница между статическим и динамическим объектом в языке программирования?
2. Как получить доступ к динамической переменной?
3. Чем определяется количество и размерность динамических переменных?
4. Приведите пример создания простой динамической перменной типа int и ее инициализации?
5. Что означает следующее описание `char *lineptr[LINES];`?

Практикум №15. СТРУКТУРЫ, ОБЪЕДИНЕНИЯ, БИТОВЫЕ ПОЛЯ

Краткие теоретические сведения

- Структуры

Структуры – это составной объект, в который входят элементы любых типов, за исключением функций. В отличие от массива, который является однородным объектом, структура может быть неоднородной. Тип структуры определяется записью вида:

```
struct { список определений }
```

В структуре обязательно должен быть указан хотя бы один компонент. Определение структур имеет следующий вид:

```
тип-данных описатель;
```

где тип-данных указывает тип структуры для объектов, определяемых в описателях. В простейшей форме описатели представляют собой идентификаторы или массивы. Пример:

```
struct
  { double x,y;
   } s1, s2, sm[9];
struct
  {  int   year;
     char  moth, day;
  } date1, date2;
```

Переменные `s1`, `s2` определяются как структуры, каждая из которых состоит из двух компонент `x` и `y`. Переменная `sm` определяется как массив из девяти структур. Каждая из двух переменных `date1`, `date2` состоит из трех компонентов `year`, `moth`, `day`. Существует и другой способ ассоциирования имени с типом структуры, он основан на использовании тега структуры. Тег структуры аналогичен тегу перечислимого типа:

```
struct тег { список описаний; };
```

где тег является идентификатором.

В приведенном ниже примере идентификатор `student` описывается как тег структуры:

```
struct student
{ char name[25];
  int  id, age;
  char  prp;
};
```

Тег структуры используется для последующего объявления структур данного вида в форме:

```
struct тег список-идентификаторов;
```

Пример:

```
struct studeut st1,st2;
```

в C++ во всех случаях использования структурированной переменной, кроме самого определения структуры служебное слово `struct` можно опускать, то есть использовать только имя структурированной переменной. Использование тегов структуры необходимо для описания рекурсивных структур. Ниже рассматривается использование рекурсивных тегов структуры:

```
struct node { int data;
              node * next; }   st1_node;
```

Тег структуры `node` действительно является рекурсивным, так как он используется в своем собственном описании, т.е. в формализации указателя `next`. Структуры не могут быть прямо рекурсивными, т.е. структура `node` не может содержать компоненту, являющуюся структурой node, но любая структура может иметь компоненту, являющуюся указателем на свой тип, как и сделано в приведенном примере.

Структуры могут быть инициализированы списками значений элементов, заключенных в фигурные скобки и перечисленных через запятую:

```
struct student st1 = { "Петров",1,20,'М' };
```

Доступ к компонентам структуры осуществляется с помощью указания имени структуры и следующего через точку имени выделенного компонента, например:

```
st1.name="Иванов";
st2.id=st1.id;
st1_node.data=st1.age;
```

Для работы с указателями на структурированные переменные существует специальная операция `->` (стрелка, минус-больше), которая понимается как выделение элемента в структурированной переменной, адресуемой указателем. Операндами здесь являются указатель на структуру и элемент структуры. Операция имеет полный аналог в виде сочетания операций `*` и `.` :

```
struct student *p,A;
```

```
p = &A;
p->age      // эквивалентно (*p).age
```

Объекты типа структур можно присваивать, передавать как параметры функции и возвращать из функции в качестве результата. Остальные операции, такие как сравнение (== и !=) не определены. Однако пользователь может определить эти операции.

Два структурных типа являются различными даже когда они имеют одни и те же члены и отличны от основных типов. Пример:

```
struct s1 { int a; };
struct s2 { int a; };
s1 x;
s2 y = x;    // ошибка: несоответствие типов
int i = x;   // ошибка: несоответствие типов
```

- Объединения (смеси)

Объединение подобно структуре, однако в каждый момент времени может использоваться только один из элементов объединения. Тип объединения может задаваться в следующем виде:

```
union {   описание элемента 1;
          ...
          описание элемента n; };
```

Главной особенностью объединения является то, что для каждого из объявленных элементов выделяется одна и та же область памяти, т.е. они перекрываются. Хотя доступ к этой области памяти возможен с использованием любого из элементов, элемент для этой цели должен выбираться так, чтобы полученный результат не был бессмысленным.

Доступ к элементам объединения осуществляется тем же способом, что и к структурам. Тег объединения может быть формализован точно так же, как и тег структуры.

Объединение применяется для инициализации используемого объекта памяти, если в каждый момент времени только один объект из многих является активным или для интерпретации основного представления объекта одного типа, как если бы этому объекту был присвоен другой тип.

Память, которая соответствует переменной типа объединения, определяется величиной, необходимой для размещения наиболее длинного элемента объединения. Когда используется элемент меньшей длины, то переменная типа объединения может содержать неиспользуемую память. Все элементы объединения хранятся в одной и той же области памяти, начиная с одного адреса. Пример:

```
union {   char  fio[30];
```

```
          char  adres[80];
          int   vozrast;
          int   telefon;   } inform;
union {   int ax;
          char al[2];      }   ua;
```

При использовании объекта `infor` типа `union` можно обрабатывать только тот элемент который получил значение, т.е. после присвоения значения элементу `inform.fio`, не имеет смысла обращаться к другим элементам. Объединение ua позволяет получить отдельный доступ к младшему `ua.al[0]` и к старшему `ua.al[1]` байтам двухбайтного числа `ua.ax`.

- Битовые поля

Элементом структуры может быть битовое поле, обеспечивающее доступ к отдельным битам памяти. Вне структур битовые поля объявлять нельзя. Нельзя также организовывать массивы битовых полей и нельзя применять к полям операцию определения адреса. В общем случае тип структуры с битовым полем задается в следующем виде:

```
struct { unsigned идентификатор 1 :  длина-поля  1;
         unsigned идентификатор 2 :  длина-поля  2;    }
```

длинна-поля задается целым выражением или константой. Эта константа определяет число битов, отведенное соответствующему полю. Поле нулевой длинны обозначает выравнивание на границу следующего слова. Пример:

```
struct { unsigned a1 :  1;
         unsigned    :  2; //неиспользуемое
         unsigned a3 :  5;
         unsigned a4 :  2;  } prim;
```

Структуры битовых полей могут содержать и знаковые компоненты. Такие компоненты автоматически размещаются на соответствующих границах слов, при этом некоторые биты слов могут оставаться неиспользованными. Допустимы неименованные поля; они не влияют на смысл именованных полей, но могут улучшить размещение.

Ссылки на поле битов выполняются точно так же, как и компоненты общих структур. Само же битовое поле рассматривается как целое число, максимальное значение которого определяется длиной поля.

- Перечисления

Существует метод определения целых констант. Например:

```
enum { A, B, C };
```

определяет три целых константы, называемы перечислителями, и присваивает им значения. Поскольку значения перечислителей по умолчанию присваиваются начиная с 0 в порядке возрастания, т.е. это эквивалентно записи:

```
const A = 0;
const B = 1;
const C = 2;
```

Перечисление может быть именованным:

```
enum key { A, B, C };
```

Имя перечисления становится синонимом `int`, а не новым типом. Например:

```
key new;
 switch (new) {
  case A:
      ...      break;
  case B:
      ...      break;
  }
```

Задавать значения перечислителей можно явно:

```
enum abc {
      a=100000,
      b=40000,
      c=1 };
```

- Определение типа данных (спецификатор typedef)

Спецификатор `typedef` позволяет в явном виде определить производный тип данных и использовать его имя в программе как обозначение этого типа, аналогично базовым (`int`, `char`...). В этом смысле он похож на определение структуры, в котором имя структуры (со служебным словом `struct`) становится идентификатором структурированного типа данных. Спецификатор `typedef` позволяет сделать то же самое для любого типа данных.

Спецификатор `typedef` имеет синтаксис контекстного определения типа данных, в котором вместо имени переменной присутствует имя вводимого типа данных. Рассмотрим пример:

```
// Синтаксис контекстного определения типа для переменной PSTR
typedef char *PSTR;    // PSTR - имя производного типа данных.
```

Тип данных `PSTR` определяется в контексте как указатель на символ (строку):

```
PSTR p,q[20],*pp;
```

Переменная p типа PSTR, массив из 20 переменных типа PSTR и указатель типа PSTR представляют собой указатель на строку, массив указателей на строку и указатель на указатель на строку.

```
long  l;
*((PSTR)&l + 2) = 5;
```

Указатель на переменную типа long преобразуется к типу указатель на строку.

С помощью typedef программисты могут устанавливать соглашения по обозначению тех или иных типов и связанных с ними структур данных. Например, тип PSTR используется для обозначения указателя на строку символов, заканчивающуюся символом '\0'. Заметим, что в этом случае никаких дополнительных действий по проверке корректности структур данных транслятором не производится: тип данных PSTR только улучшает читаемость программы.

Задания для выполнения

Таблица 23

Определить структурированный тип, набор функций (в виде меню) для работы с массивом структур. В структурированной переменной предусмотреть способ отметки ее как не содержащей данных (т.е. «пустой»). Функции должны работать с массивом структур или с отдельной структурой через указатели, а также при необходимости возвращать указатель на структуру. В перечень обязательных функций входят: **- «очистка» структурированных переменных;** **- поиск свободной структурированной переменной;** **- ввод элементов (полей) структуры с клавиатуры;** **- вывод элементов (полей) структуры с клавиатуры;** **- поиск в массиве структуры и минимальным значением заданного поля;** **дополнительные функции** **- сортировка массива структур в порядке возрастания заданного поля (при сортировке разрешается присваивание структурированных переменных);** **- удаление заданного элемента;** **- изменение (редактирование) заданного элемента.** **Интерфейс пользователя осуществить в виде командного процессора:** **1 - загрузить данные** **2 - вывести на экран .....**	
1	Горожанин. Фамилия И.О., дата рождения, адрес. Выбор по произвольному шаблону.
2	База данных гостиница: Номер страницы, номер строки, текст изменения строки, дата изменения. Выбор по произвольному шаблону.

Продолжение таблицы 23

3	Клиенты банка. Фамилия И.О., номер счета, сумма на счете, дата последнего изменения. Выбор по произвольному шаблону.
4	Читатели. Фамилия И.О., номер читательского билета, название книги, срок возврата. Выбор по произвольному шаблону.
5	База данных студент. Фамилия И.О., номер зачетной книжки, факультет, группа. Выбор по номеру зачетной книжки.
	Склад. Наименование товара, цена, количество, процент торговой надбавки. Выбор по произвольному шаблону.
6	База данных авиарейсов. Номер рейса, пункт назначения, время вылета, дата вылета, стоимость билета. Выбор по произвольному шаблону.
7	Ученик. Фамилия И.О., количество оценок, оценки, средний балл. Выбор по фамилии.
8	База данных студент. Фамилия И.О., дата поступления, дата отчисления. Выбор по произвольному шаблону.
9	База данных автосервис. Регистрационный номер автомобиля, марка, пробег. Выбор по произвольному шаблону.
10	Телефонная станция. Фамилия И.О., количество переговоров (для каждого - дата и продолжительность). Выбор по фамилии.
11	Телефонная станция-2. Номер телефона, дата разговора, продолжительность, код города. Выбор по номеру телефона.
12	Вокзал. Номер поезда, пункт назначения, дни следования, время прибытия, время стоянки. Выбор по произвольному шаблону.
13	Кинотеатр. Название кинофильма, сеанс, стоимость билета, количество зрителей. Выбор по назаванию конофильма.
14	Преподаватели. Название экзамена, дата экзамена, фамилия преподавателя, количество оценок, оценки. Выбор по произвольному шаблону.
15	Отдел кадров. Паспортные данные, образование, специальность, подразделение, должность, оклад, даты поступления в фирму и последнего назначения и т. д. Выбор по произвольному шаблону.
16	Справочник начальника тюрьмы. Анкетные данные заключенных, статья, срок, дата заключения под стражу, место в тюремной иерархии, камера, сведения о родственниках, особенности характера. Формирование статистических сводок о составе.
17	Справочник коммерческих банков. Наименование, адрес, статус (форма собственности), условия хранения средств на лицевом счете (годовые проценты на различных видах вкладов). Выбор банка с наибольшим процентом для заданного типа вклада.
18	Склад. Наименование, единица измерения, цена единицы, количество, дата последнего завоза. Регистрация поступления товара (формирование приходной накладной) и отгрузки (расходная накладная).

Продолжение таблицы 23

19	Личная библиотека. Картотека домашней библиотеки: выходные данные книги (авторы, название, издательство и так далее), раздел библиотеки (специальная литература, хобби, домашнее хозяйство, беллетристика и так далее), происхождение и наличие книги в данный момент. Выбор книг по произвольному запросу.
20	Администратор гостиницы. Список номеров: класс, число мест. Список гостей: паспортные данные, даты приезда и отъезда, номер. Поиск гостя по произвольному признаку.
21	Справочник лекаря. База болезней: название, симптомы, процедуры, перечень рекомендуемых лекарств с указанием требуемого количества. . Формирование рецепта после осмотра больного.
22	Справочник лекаря-2. База медикаментов на складе: название, количество, взаимозаменяемость, цена. Корректировка запасов.
23	Справочник ГИБДД. Марка, цвет, заводской и бортовой номера, дата выпуска, особенности конструкции и окраски, дата последнего техосмотра транспортного средства, паспортные данные владельца. Выбор транспортных средств по произвольному шаблону.
24	Записная книжка. Анкетные данные, адреса, телефоны, место работы или учебы, должность знакомых, коллег и родственников, характер знакомства, деловые качества и так далее. Поиск по произвольному шаблону.
25	Шеф-повар. База рецептур блюд: а) раскладка и название (3-4 компонента), б) база продуктов на складе: наименование, цена, количество. Формирование меню на день (на заданное число персон); званый ужин.
26	Бюро знакомств. База потенциальных женихов и невест: пол, регистрационный номер, дата регистрации, сведения о себе, требования к партнеру. Выбор подмножества подходящих кандидатур.
27	Картотека Интерпола. Данные по каждому зарегистрированному преступнику: фамилия, имя, кличка, рост, цвет волос и глаз, особые приметы, гражданство, место и дата рождения, последнее место жительства, знание языков, преступная профессия, последнее дело и так далее. Выборка по любому подмножеству признаков.
28	Зачисление абитуриентов. Анкетные данные, совокупность оценок на вступительных экзаменах, готовность учиться на договорной основе. Выбор для зачисления заданного количества абитуриентов.
29	Справочник абитуриента. База вузов: наименование, адрес, перечень специальностей, конкурс прошлого года по каждой специальности (дневной, вечерней, заочной форм), размер оплаты при договорном обучении. Выбор по разным критериям: все о данном вузе; все о данной специальности.
30	Ломбард. База хранимых товаров и недвижимости: анкетные данные клиента, наименование товара, оценочная стоимость; сумма, выданная под залог, дата сдачи, срок хранения. Операции продажи по истечении срока хранения

Контрольные вопросы

1. С помощью `typedef` определите типы: беззнаковый `char`; константный беззнаковый `char`; указатель на целое; указатель на указатель на `char`; вектор из 7 целых указателей.

2. Определите массив структур.

3. Что является левым операндом операции `(.)` при обращении к полю структуры?

4. Истинно ли следующее утверждение: вы можете присвоить значение одной структурной переменной другой структурной переменной того же типа?

5. Что объединяет перечисление: данные различных типов; логически связанные переменные; именованные целые числа; константные значения?

Практикум № 16-17. РАБОТА С ФАЙЛАМИ

Краткие теоретические сведения

Особенностью С является отсутствие в этом языке структурированных файлов. Все файлы рассматриваются как не структурированная последовательность байтов. При таком подходе понятие файла распространяется и на различные устройства.

Библиотека поддерживает три уровня ввода-вывода: потоковый ввод-вывод; ввод-вывод нижнего уровня; ввод-вывод для консоли и портов (зависит от операционной системы – ОС).

- Потоковый ввод-вывод

На уровне потокового ввода-вывода обмен данными производится побайтно, т. е. за одно обращение к файлу производится считывание или запись фиксированной порции данных (512 ил 1024 байта). При вводе с диска или при считывании из файла данные помещаются в буфер ОС, затем побайтно или порциями передаются в программе пользователю. При вывод в файл данные накапливаются в буфере, а при заполнении буфера записываются в виде единого блока на диск. Буферы ОС реализуются в виде участков основной памяти.

Поток – это файл вместе с предоставленными средствами буферизации. Потоки можно открывать и закрывать (связывать указатели на поток с конкретными файлами); вводить и выводить строку, символ, форматированные данные, порцию данных произвольной длины; анализировать ошибки ввода-вывода и достижения конца файла; управлять буферизацией потока и размером буфера; получать и устанавливать указатель текущей позиции в файле.

Прежде чем начать работать с потоком, его надо инициировать, (открыть). При этом поток связывается со структурой предопределенного типа `FILE`, определение которой находится в библиотечном файле `<stdio.h>`. В структуре находится указатель на буфер, указатель на текущую позицию файла и т. п. При открытии потока, возвращается указатель на поток, т. е. на объект типа `FILE`:

```
#include <stdio.h>;
. . .
FILE *fp;
. . .
fp= fopen( ”t.txt”, ”r”);
```

где `fopen(<имя_файла>,<режим_открытия>)` – функция для инициации файла.

В стандартной библиотеке каждый файл представлен структурированной переменной, в которой сосредоточена вся информация об открытом файле: тип, идентификатор файла в операционной системе (номер, handle), буфер на 1 блок (сектор), текущее состояние и способ работы с файлом:

```
typedef struct {.....} FILE;
```

Часто ее называют описателем или дескриптором файла. При открытии файла функция `fopen` создает переменную – дескриптор файла и возвращает указатель на нее. Программа должна его запомнить и в дальнейшем использовать при всех обращениях к файлу для его идентификации.

В табл. 24 приведены режимы для открытия файла.

Таблица 24

Режим	Описание
″w″	открыть файл для записи, если файл существует, то он стирается
″r″	открыть файл для чтения
″a″	открыть файл для добавления, если файл существует, то он не стирается и можно писать в конец файла
″w+″	открыть файл для записи и исправления, если файл существует, то он стирается, а далее можно и читать , и писать, размеры файла можно увеличивать
″r+″	открыть файл для чтения и записи, но увеличить размер файла нельзя
″a+″	открыть файл для добавления, т. е. можно и читать и писать, в том числе и в конец файла

Поток можно открыть в текстовом (t) или двоичном (b) режиме. Текстовые файлы являются по своей природе файлами последовательного доступа. Двоичный файл выступает как аналог внутренней памяти компьютера и организован соответствующим образом. По умолчанию устанавливается текстовый режим. В явном виде режим указывается следующим образом: `″r+b″` или `″rb″` – двоичный (бинарный) режим. Пример:

```
if ((fp=fopen(”t.txt”, ”w”)==NULL)
{
perror(”\nошибка при открытии файла”);
        // выводит строку символов с сообщением об ошибке
exit(0);
}
        // после работы с файлом, его надо закрыть
fclose(<указатель_на_поток>);
```

- Блоковый ввод-вывод

Для блокового ввода и вывода используются функции:

```
int fread( void *ptr, int size, int n, FILE *fp);
```

где `void *ptr` – указатель на область памяти, в которой размещаются считываемые из файла данные; `int size` – размер одного считываемого элемента; int n - количество считываемых элементов; `FILE *fp` – указатель на файл, из которого производится считывание. В случае успешного считывания информации функция возвращает число прочитанных элементов (а не байтов), иначе – EOF.

```
int fwrite( void *ptr, int size, int n, FILE *fp);
```

где `void *ptr` – указатель на область памяти, в которой размещаются записываемые в файл данные; `int size` – размер одного записываемого элемента; `int n` – количество записываемых элементов; `FILE *fp` – указатель на файл, в который производится запись. В случае успешной записи информации функция возвращает число записанных элементов, иначе – EOF. Пример:

```
typedef STRUCT
{
  char name [40];
  char post [40];
  float rate;
}EMPLOYEE;
void main ()
{
FILE *f;                // указатель связанный с файлом
EMPLOYEE e;             // переменная
EMPLOYEE mas[10]        //массив
                        //открываем файл
if ((f=fopen("f.dat", "wb")==NULL) exit(1);
// если при открытии файла возникает ошибка, то выходим из функции
int i;
  for(i=1; i<=10;i++)
   {
                        //формируем запись e
    printf("name="); scanf("%s",&e.name);
    printf("post="); scanf("%s",&e.post);
    printf("rate="); scanf("%f",e.rate);
                        // записываем запись e в файл
    fwrite(&e, sizeof(EMPLOYEE),1,f);
    if (ferror(f)==NULL) exit(2);
   }
fclose(f);
                        //чтение записей из файла
if ((f=fopen("f.dat", "rb")==NULL) exit(3);
// если при открытии файла возникает ошибка, то выходим из функции
i=0;
```

```
while(!feof(f)&&i<=10)
  {
    fread(&mas[i], sizeof(EMPLOYEE),1,f);
    i++;
   }
fclose(f);
}
```

- Строковый ввод-вывод

Для построчного ввода-вывода используются следующие функции:

```
char *fgets(char *s, int n, FILE *F);
```

где `char *s` – адрес, по которому размещаются считанные байты; `int n` – количество считываемых байтов; `FILE *fp` – указатель на файл, из которого производится считывание.

Прием символов заканчивается после передачи `n` байтов или при получении "\n". Управляющий символ "\n" тоже передается в принимающую строку. В любом случае строка заканчивается "\0". При успешном завершении считывания, функция возвращает указатель на прочитанную строку, иначе возвращает NULL.

```
char *fputs(char *s, FILE *F);
```

где `char *s` – адрес, из которого берутся записываемые в файл байты; `FILE *fp` – указатель на файл, в который производится запись. Пример:

```
int MAXLINE=255; //максимальная длина строки
FILE    *in,     //исходный файл
*out;            //принимающий файл
char buf[MAXLINE];//строка, с помощью которой выполняется копирование
//копирование строк одного файла в другой
while (fgets (buf, MAXLINE, in)!=NULL)
fputs(buf,out);
```

- Поизиционирование в файле

С открытым файлом связано понятие «текущей позиции» (позиционера). Текущая позиция – номер байта, начиная с которого производится очередная операция чтения-записи. При открытии файла текущая позиция устанавливается на начало файла, после чтения-записи порции данных перемещается вперед на размерность этих данных. Для дополнения файла новыми данными необходимо установить текущую позицию на конец файла и выполнить операцию записи. Текущая позиция представляется в программе переменной типа `long`. Для работы с ней в стандартной библиотеке имеются две функции:

```
long  ftell(FILE *fp);
```

возвращает текущую позицию в файле. Если по каким-то причинам текущая позиция не определена, функция возвращает `-1L`. Вторая функция устанавливает текущую позицию в файле на байт с номером `pos`:

```
int   fseek(FILE *fp, long pos, int mode);
```

Параметр `mode` определяет, относительно чего отсчитывается текущая позиция в файле, и имеет следующие символические и числовые значения (установленные в `stdio.h`):

```
#define     SEEK_SET 0  // Относительно начала файла – позиция 0
#define     SEEK_CUR 1  // Относительно текущей позиции,
                        // >0 - вперед, <0 - назад
#define     SEEK_END 2  // Относительно конца файла
                        // (значение pos - отрицательное)
```

Функция `fseek` возвращает значение 0 при успешном позиционировании и -1 (EOF) – при ошибке. Получить текущую длину файла можно позиционированием:

```
long  fsize;
fseek(fl,0L,SEEK_END);  // Установить позицию на конец файла
fsize = ftell(fd);      // Прочитать значение текущей позиции
```

Задания для выполнения

Таблица 25

Сформировать двоичный файл из элементов, заданной в варианте структуры, распечатать его содержимое, выполнить удаление и добавление элементов, используя для поиска удаляемых или добавляемых элементов функцию. Формирование, печать, добавление и удаление элементов оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.			
1	«Абитуриент»: фамилия, имя, отчество; год рождения; оценки вступительных экзаменов; средний балл аттестата. Удалить элемент с указанным номером, добавить элемент после элемента с указанной фамилией.	2	«Сотрудник»: фамилия, имя, отчество; должность, год рождения; заработная плата. Удалить элемент с указанной фамилией, добавить элемент после элемента с указанным номером.
3	«Государство»: название; столица; численность населения; занимаемая площадь. Удалить все элементы, у которых численность меньше заданной, добавить элемент после элемента с указанным номером.	4	«Человек»: фамилия, имя, отчество; домашний адрес; номер телефона; возраст. Удалить все элементы с заданным возрастом, добавить элемент после элемента с заданным номером.

Продолжение таблицы 25

5	«Информация»: носитель; объем; название; автор. Удалить первый элемент с заданным объемом информации, добавить элемент перед элементом с указанным номером.	6	«Школьник»: фамилия, имя, отчество; класс; номер телефона; оценки по предметам (4). Удалить все элементы, у которых есть 2 хотя бы по одному предмету, добавить элемент в начало файла.
Выполнить задание (необходимые файлы создаются программно)			
7	Компоненты файла *f* – вещественные числа (положительные и отрицательные). Определить и вывести на экран порядковый номер того из них, которое наиболее близко к какому-либо целому числу.		
8	Компоненты файла *f* – целые (отличные от нуля) числа, причем 10 положительных чисел, 10 отрицательных, и т.д. Получить файл *g,* в котором записаны сначала 5 положительных чисел, затем 5 отрицательных и т.д.		
9	Компоненты файла *f* – целые числа. Получить файл *g*, образованный из *f* исключением повторных вхождений одного и того же числа.		
10	Дан строковый файл с именем *NameS*, содержащий даты в формате "день/месяц/год", причем под день и месяц отводится по две позиции, а под год – четыре. Создать файлы целых чисел с именами *Name1* и *Name2*, содержащие соответственно значения [дней и месяцев]; [дней и лет]; [месяцев и лет] для дат из исходного строкового файла (в том же порядке).		
11	Даны три файла целых чисел одинакового размера с именами *NameA, NameB* и *NameC*. Создать новый файл с именем *NameD*, в котором чередовались бы элементы исходных файлов с одним и тем же номером: *A0*, *B0*, *C0*, *A1*, *B1*, *C1*, *A2*, *B2*, *C2*, ...		
12	Дан строковый файл, содержащий даты в формате "день/месяц/год", причем под день и месяц отводится по две позиции, а под год — четыре. Создать новый строковый файл, в котором даты из исходного файла располагались бы в порядке возрастания.		
13	Дан файл вещественных чисел, содержащий элементы прямоугольной матрицы (по строкам), причем начальный элемент файла содержит количество столбцов матрицы. Создать новый файл той же структуры, содержащий матрицу, транспонированную к исходной.		
14	Даны два файла вещественных чисел с именами *NameA* и *NameB*, содержащие элементы прямоугольных матриц *A* и *B* (по строкам), причем начальный элемент каждого файла содержит количество столбцов соответствующей матрицы. Создать файл той же структуры с именем *NameC*, содержащий произведение *A·B*. Если матрицы *A* и *B* нельзя перемножать, то оставить файл *NameC* пустым.		
15	Дано целое число *N* (< 5) и *N* файлов одного и того же типа с именами *Name1, ..., NameN*. С помощью процедур *BlockRead* и *BlockWrite* объединить содержимое этих файлов (в указанном порядке) в новом файле с именем *Name0.*		

Продолжение таблицы 25

16	Даны два файла вещественных чисел с именами *Name1* и *Name2*, элементы которых упорядочены по возрастанию. Объединить эти файлы в новый файл с именем *Name3*, сохранив упорядоченность элементов.
17	Дан строковый файл, содержащий непустые строки. Создать новый файл, содержащий все строки исходного файла наибольшей длины (в том же порядке).
18	Дано число *k* и строковый файл с именем *Name1*, содержащий непустые строки. Создать два новых файла: строковый с именем *Name2*, содержащий последние *k* символов каждой строки исходного файла (если строка короче *k* символов, то она сохраняется целиком), и символьный с именем *Name3*, содержащий *k*-й символ каждой строки (если строка короче *k*, то в файл *Name3* записывается пробел).
19	Даны четыре файла целых чисел разного размера с именами *NameA*, *NameB*, *NameC* и *NameD*. Создать новый файл *NameE*, в котором чередовались бы элементы исходных файлов с одним и тем же номером. «Лишние» элементы более длинных файлов в результирующий файл не записывать.
20	Дан строковый файл, содержащий даты в формате "день/месяц/год", причем под день и месяц отводится по две позиции, а под год – четыре. Вывести строку, содержащую самую позднюю весеннюю|летнюю|осеннюю|зимнюю дату. Если даты с требуемым временем года в файле отсутствуют, то вывести дату «01/01/1900».
Создать текстовый файл *F1* не менее, чем из 10 строк и записать в него информацию. Выполнить задание	
21	Скопировать в файл *F2* только четные строки из *F1*. Подсчитать размер файлов *F1* и *F2* (в байтах).
22	Скопировать в файл *F2* только те строки из *F1*, которые начинаются с буквы «А». Подсчитать количество слов в *F2*.
23	Скопировать из файла *F1* в файл *F2* строки, начиная с *K* до *K+5*. Подсчитать количество гласных букв в файле *F2*
24	Скопировать из файла *F1* в файл *F2* все строки, которые не содержат цифры. Подсчитать количество строк, которые начинаются на букву «А» в файле *F2*.
25	Скопировать из файла *F1* в файл *F2* все строки, которые содержат только одно слово. Найти самое длинное слово в файле *F2*.
Выполнить задания с использованием текстового файла	
26	Проверить файл на соответствие числа открывающих и закрывающих скобок комментариев (круглых со звездочкой и фигурных) и вывести в файл тексты комментариев.
27	Определить функцию, которая, игнорируя исходное деление файла на строки, переформатирует его, разбивая на строки так, чтобы каждая строка оканчивалась «;», или содержала 60 символов, если среди них нет «;».
28	Составить программу, которая собирает информацию обо всех циклах с указанием номеров строк начала и конца каждого цикла и его типа.

Окончание таблицы 25

29	В последовательном текстовом файле построчно занесены до 50 целых положительных чисел (каждое число <10000). Каждое новое число – в новой строке. Получить файл, в котором все числа исходного файла расположены по возрастанию без повторений.
30	Написать функции, обеспечивающие шифровку и расшифровку текста, записанного файле и состоящего из русских заглавных и строчных букв, а также знаков препинания. Для шифрования текста записать его в квадратную матрицу наименьшего подходящего размера по столбцам, а затем прочитать его по строкам.
Указанный вариант заданий реализовать с использованием позиционирования указателя в текстовом файле и массива указателей, без загрузки самого текстового файла в память	
31	Сортировка строк файла по длине и по алфавиту и вывод результата в отдельный файл
32	Программа – редактор текста с командами удаления, копирования, и перестановки строк, с прокруткой текста в обоих направлениях (исходный файл при редактировании не меняется
33	Программа просмотра блочной структуры С-программы с командами вывода текущего блока, входа в *n*-ый по счету вложенный блок и выхода в блок верхнего уровня.
34	Программа построчного сравнения двух файлов с выводом групп строк, вставленных или удаленных из второго файла относительно первого.
35	Программа просмотра текстового файла по предложениям. Предложением считается любая последовательность слов, ограниченная точкой, после которой идет большая буква или конец строки. Программа выводит на экран любой блок с *n*-го по m-ое предложение.
36	Программа сортировки файла по длине предложений и вывода результата в отдельный файл. При выводе каждое предложение следует переформатировать так, чтобы оно начиналось с отдельной строки и располагалось в строках размером не более 60 символов.

Контрольные вопросы

1. Что такое дескриптор файла?
2. Определить понятие файл последовательного доступа и перечислить функции для работы сним.
3. Определить понятие двоичный файл произвольного доступа.
4. Что такое запись для файла?
5. Как размещаются в файл данных динамические структуры?

ОГЛАВЛЕНИЕ

Учебное издание

Пацей Наталья Владимировна
Занько Дмитрий Владимирович

КОНСТРУИРОВАНИЕ ПРОГАММ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Практикум

Редактор

Подписано в печать 22.09.2005. Формат 60x84 1/16
Бумага офсетная. Гарнитура Таймс. Печать офсетная.
Усл. печ. л. 6,2. Уч.-изд. л. 6,4.
Тираж 70 экз. Заказ

Учреждение образования «Белорусский государственный технологический университет». 220050. Минск, Свердлова, 13а.
ЛИ № 02330/0133255 от 30.04.2004.

Отпечатано в лаборатории полиграфии учреждения образования «Белорусский государственный технологический университет». 220050. Минск, Свердлова, 13.
ЛП № 02330/0056739 от 22.01.2004.