Министерство образования Российской Федерации

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет математики и информатики

Т. А. Галаган, Л.А. Соловцова

ЯЗЫК ПРОГРАММИРОВАНИЯ С++ В ПРИМЕРАХ И ЗАДАЧАХ.

Учебно-методическое пособие

Благовещенск
2002

ББК 32.973–018я73
Г15

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Галаган Т.А., Соловцова Л.А.

Язык программирования С++ в примерах и задачах: Учебно–методическое пособие по дисциплине "Алгоритмические языки и программирование" для студентов специальностей 220200 и 071900 очной формы обучения. Благовещенск: Амурский гос. ун-т, 2002.

Пособие посвящено знакомству с популярным языком программирования С+ +. Содержит множество простых для понимания примеров. В каждом из разделов имеются задания для быстрой самопроверки и перечень задач по программированию. Пособие предназначено для студентов, но может быть интересно и преподавателям.

Т.А. Галаган подготовлены разделы: указатели, ссылки, файлы, классы; Л.А Соловцовой – функции, массивы, структуры и объявления.

.

Рецензент: *А.П. Докучаев, ведущий инженер-программист*
ОАО «Санго-Плюс»

ВВЕДЕНИЕ

Пособие посвящено знакомству с популярными языками программирования С и С++. В основу их положено значительно меньше синтаксических правил, чем у других языков программирования. Язык менее строго структурирован и предоставляет программисту свободу выбора альтернативных решений. Многие программисты, впервые столкнувшись с ним, находят его слишком замысловатым и пугающим. Конструкции языка, напоминающие выражения на английском языке и характерные для многих языков программирования, в С встречаются довольно редко. Вместо этого программист сталкивается с необычного вида операторами и обилием указателей. Но программы, написанные на С и С+ +, отличаются высокой эффективностью и возможностью создавать системные приложения за счет управления переменными на битовом уровне.

Необходимо подчеркнуть, что пособие не претендует на полное освещение основ программирования на языках С и С+ +. Оно предполагает, что читатель знаком с такими понятия языка как типы переменных, их объявление и основные операции. Пособие служит дополнительным материалом при проведении практических занятий по дисциплине «Алгоритмические языки и программирование» у студентов I курса специальностей 220200 «Автоматизированные системы обработки информации и управления» и 071900 «Информационные системы». Материал разбит на следующие разделы: «Функции», «Массивы», «Указатели», «Ссылки», «Структуры и объединения», «Файлы», «Классы». В каждом из них приводится краткая справочная информация по теме. В пособии прежде всего рассматриваются примеры программ, которые можно набирать и запускать в среде Borland С+ + версии 3.1 .

Все разделы содержат также контрольные вопросы, дающие возможность быстрой проверки, насколько начинающий программист усвоил основные понятия и навыки программирования. Вопросы допускают только однозначные ответы и подготовлены таким образом, чтобы на каждый из них можно было ответить за несколько минут. Правильность ответа можно проверить, заглянув в конец пособия.

Задачи для программирования, предлагаемые в разделах, требуют разработки законченной программы.

Необходимо отметить, что пособие не претендует на полное освещение языка С+ +, хотя и использует С+ + как средство обучения концепциям программирования.

ФУНКЦИИ

Любая программа, написанная на языке C++, есть последовательность выполнения функций, причем одна из них обязательно должна носить имя main( ). Выполнение программы всегда начинается с выполнения ряда операторов этой функции. Все функции в языке C++ равноправны: каждая из них (даже main( )) может быть вызвана любой другой функцией. Функция также может вызывать саму себя (явление рекурсии). Описание функции состоит из заголовка и тела.

Заголовок <тип результата> main (список аргументов) имя функции
{*Тело функции*}

Если тип результата не указывается, то предполагается, что функция возвращает значение типа int. Если функция не возвращает результат, указывается слово void. Наличие списка аргументов не обязательно, но круглые скобки всегда должны присутствовать после имени функции.

Пример 1

```
//Программа содержит пример функции, не принимающей никаких аргументов
#include <stdio.h>
#include <math.h>          //подключение библиотеки математических функций
void outsqr (void)
{
 int it;
 printf("Введите целое число");
 scanf( "%d", &it);
double du;
du = sqrt(it);
printf ("Квадратный корень числа %d равен %f \n", it, du);
}
main ( )
{
printf ("Эта программа вычисляет квадратный корень целого числа. \n");
outsqr ( );
}
```

Часто употребляется описание типов аргументов сразу при их объявлении внутри круглых скобок. Если аргументов несколько, их описания (тип и имя) разделяются запятыми. Если функции не передаются величины, то вместо списка аргументов необходимо задавать ключевое слово void. Локальные переменные, отличные от аргументов, описываются внутри тела функции.

Возвращает значение оператор return, с помощью которого можно передать в вызывающую функцию только одно значение.

Пример 2

Опишем функцию, возвращающую среднее значение трех величин.

```
// определение функции Average
```

```
long Average (long val1, long val2, long val3 )
{
   long sum = val 1 + val 2 + val 3;
   return sum / 3;
}
```

Пусть в программе объявлены некоторые переменные: long a, a1, a2, a3; тогда возможен вызов описанной в примере функции в виде:
a = Average (a1, a2, a3); или a = Average (525,675,819);

В случаях, когда вызываемая функция не возвращает значение, ее вызов представляет собой просто имя функции, а в круглых скобках – фактические значения аргументов функции.

Пример 3

```
//Программа, которая по введенным числам выводит на экран значения их
//кубов.
#include <iostream>
void square (float x)        //описание функции square
{
  float y=x*x*x;
  cout<<”Квадрат ”<<x<<”равен”<<y<<endl;
}

void main( void)             //описание функции main
{
   int k;
   float value;
   cout<<”Введите количество чисел”<<endl;
   cin>>k;
   for (int i=1; i<=k; i++)
   {
         cout<<”Введите число”<<endl;
         cin>>value;
         square(value);
   }
}
```

Нередким бывает случай, когда функция вызывается до того, как будет объявлена. Тогда используется прототип функции, который информирует компилятор о существовании функции, о типе возвращаемого значения, а также о типе и количестве аргументов, которые ей передаются.

Прототип функции имеет следующий вид:
< возвращаемый тип > имя функции (параметры);

В списке параметров обычно указываются тип и имя для каждой переменной; элементы списка разделяются запятыми. Указание имени переменной в прототипе не обязательно, но, как правило, применяется. Так, для функций из примеров прототип выглядит следующим образом:

long Average (long val1, long val2, long val3);
void square (float x);

Прототипы функций, как правило, располагают после директив препроцессора, до описания основной функции.

В языке С++ возможен рекурсивный вызов функции, причем количество рекурсивных вызовов не ограничивается, а определяется лишь возможностями компьютера.

Пример 4

```
// Программа, демонстрирующая применение рекурсии при вычислении
// факториала.
#include <stdio.h>
double factorial (int number); // объявление прототипа функции factorial
int main( )
{
   int n;
   double result;
   printf (“Введите число\n”);
   scanf (“%d”,&n);
   result=factorial(n);
   printf(“факториал %d равен %15.0f\”, n, result);
   return (0);
}
double factorial (int number);   // описание функции factorial
{
   if (number<=1)          return (1.0);
   else return(number*factorial(number – 1));
}
```

Быстрая самопроверка

1. Функция с именем prim является не возвращающей значение и имеет два аргумента целого типа. Какой из вариантов прототипа функции правильный:
   a) prim (int x, int y); **b)** void prim (int x, y);
   c) void prim (int , int); **d)** void prim(int x; int y);

2. Объявлен прототип функции – int mean (int a, int &b);
и переменные – int x, y, z=5; float e;
Какой из вариантов вызова функции будет верным:
a) mean(x, y); **b)** y=mean(z, x); **c)** y=mean(x, z); **d)** e=mean(x, y);

3. Найдите ошибки в описании функции:
   function (int x, k) : float
   float z=5, q;

```
{   z+=(x*x)/(2+q);
 return z;
}
```

Задачи по программированию

1. Определить периметры двух треугольников, заданных координатами их вершин. Длину стороны треугольника вычислять в функции.
2. Вычислить среднее арифметическое объемов шаров с радиусами r_1, r_2, r_3. Для нахождения объема шара использовать функцию.
3. Вычислить значение $z = (sign\, x + sign\, y)\, sign(x + y)$,

   где $sign\, a = \begin{cases} -1, & при \quad a < 0, \\ 0, & при \quad a = 0, \\ 1, & при \quad a > 0. \end{cases}$
4. Составить программу вычисления значений:

   $$a = \frac{\sqrt{z^3 + 4z^2 + 7z + 1}}{2 + e^{2z^3 + z - 3}}, \quad b = \frac{t^2 + 13t + 16}{7t^3 + t^2 - 4}.$$

   Для определения значений многочленов использовать функцию. Значения величин z, t вводить с клавиатуры.
5. Вычислить значение $z = \underbrace{\sqrt{1+x\sqrt{1+x\sqrt{1+x...}}}}_{n\ раз}$ для любого значения n.

МАССИВЫ

Для создания массива компилятору необходимо знать тип данных, количество элементов в массиве и требуемый класс памяти. Массивы могут иметь те же типы данных, что и идентификаторы, что и простые переменные.

Синтаксис объявления массива следующий:

<Тип данных> <имя массива> [размерность массива];

Например:

```
int array[45];
double rt[12];
const int ARRAY_SIZE=90;
float alp[ARRAY_SIZE];
```

Нумерация элементов массива в языке С++ начинается с нуля. Возможна также инициализация массива при его объявлении. Для этого непосредственно после объявления массива в фигурных скобках через запятую перечисляются значения элементов массива. Количество элементов должно соответствовать размеру массива. Если элементов окажется меньше, то недостающие элементы будут инициализированы нулями, если же элементов окажется больше – компилятор выдаст ошибку. При инициализации допускается использование пус-

тых скобок в объявлении массива, и тогда компилятор сам определяет размерность массива.

Чтобы получить доступ к отдельным элементам, после имени массива записывается индекс нужного элемента, заключенный в квадратные скобки:

```
array[0]=56;
array[2]=7*array[0]+3;
```

Пример 1

```
//Программа выдает количество дней в месяце для невисокосного года.
include<stdio.h>
int days[]={31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
void main ( )
{
for (int index=0; index<sizeof(days)/(sizeof(int)); index++)
printf("месяц %d имеет %d дней \n", index+1,days[index]);
}
```

В данном примере используется операция sizeof( ), определяющая размер памяти в байтах, который соответствует объекту или типу, следующему за ней в скобках.

Для объявления многомерного массива необходимо после его имени задать несколько размеров, заключенных в скобки. Размерность массивов не ограниченна. Двумерный массив в С++ представляется как массив, элементами которого являются одномерные массивы. Например:

```
float dam[4][5];
```

В этом случае массив будет содержать 4 строки и 5 столбцов. Возможна инициализация двумерного массива при объявлении, которая происходит следующим образом:

```
float art [5] [2] =    {       {1.2, 1.5},
                               {-4.0,  3.6},
                               {2.3, - 6.1},
                               {7.3,   0.4},
                               {0.0, - 2.7}   };
```

При этом внутренние фигурные скобки могут быть опущены.

Пример 2

```
// Программа, находящая разность максимального и минимального элементов в
// каждой строке матрицы размера 5x4.
#include<iostream.h>
void main( )
{ int matr [5][4]= {1, 2, 52, -4, 0, -33, 67, -2, 3, -36, 1, 37, 3, 0, 49, 10, 0, -2, 7, 15};
  int i, j, min, max;
  for (i=0; i<5; i++)
       { min=matr[j][0]; min=matr[j][0];
         for (j=0; j<4; j++)
              {
```

```
        if (min>matr[i][j]) min=matr[i][j];
        if (max<matr[i][j]) max=matr[i][j];
        }
    razn=max−min;
    cout<<"разность в "<<i<<"строке  равна "<<razn<<endl;
    }
}
```

При разработке программного обеспечения очень распространенной операцией является сортировка значений в списках. Сортировка − расположение списка в некотором порядке (например, слова в алфавитном порядке или значения − в возрастающем или убывающем порядке). Одним из методов сортировок является сортировка методом прямого выбора, при котором последовательно происходит просмотр всего списка значений, выбор из него минимального или максимального (в зависимости от порядка сортировки) и расположение его на нужном месте, обмен местами с элементом, стоявшим там ранее. Данный алгоритм представлен на рис.1.

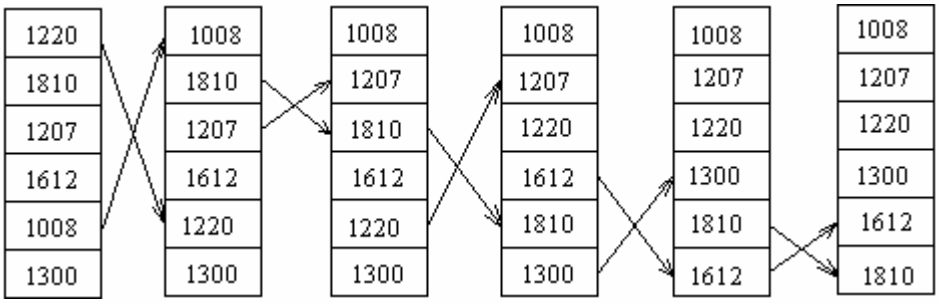

Рис.1 Сортировка методом прямого выбора

Для программной реализации данного алгоритма необходимо введение некоторой временной переменной, чтобы при обмене значения переменных не терялись. Функция, с помощью которой реализуется сортировка методом прямого выбора, приведена ниже.

Пример 3

```
void Sort ( int list[ ], int length)
// сортирует элементы массива по возрастанию
{
 int temp;                  //переменная для временного хранения
 int passCount;             //переменная управления циклом
 int place Count;           //переменная управления циклом
 int minIndex;              //индекс минимального значения
    for (passCount=0; passCount<length−1; passCount++)
   {
        minIndex=passCount;
//ищем индекс минимального элемента среди значений list[passCount…length-1]
        for (placeCount = passCount+1; placeCount< length; placeCount++)
              if ( list[placeCount]<min[minIndex] )  minIndex=placeCount;
```

```
        //меняются местами list[minIndex] и list[passCount]
        temp=list[minIndex];
        list[minIndex]=list[passCount];
        list[passCount]=temp;
    }
}
```

Чтобы произвести сортировку по убыванию, нужно каждый раз искать вместо минимального значения максимальное.

Быстрая самопроверка

1. Объявление массива типа float с именем plan, имеющего 30 строк и 10 столбцов, выглядит как

a) float plan[10][30]; b)float plan [29][19]; c)float plan [31][11];
d) float plan [30,10]; e)float plan [30][10];

2. Массив инициализирован при объявлении следующим образом:
int array[]={1, 6, 8, 2, 9, 4, 0};
Значение элемента array[5] равно

a) 9 b) 4 c) 2

3. Массив инициализирован как:
float matrix [4] [2] ={{1.2, 6.5}, {-4.0, 9.6}, {2.3}, {0.0, -2.7}};
Значение 0.0 имеет элемент

a) matrix[3][1] b) matrix[2][1] c) matrix[3][0] d) matrix[3][1]

4. Даны объявления: int sample[8], k;
Выберите содержимое массива sample после следующего выполнения программного кода:

```
for (k=0; k<8; k++)
if ( k%2= =0 )      sample[k]=k;
else                sample[k]= 10 - k;
```

a) 0 9 8 7 4 5 6 3 b) 0 9 8 7 6 5 4 3 c) 0 9 1 8 3 7 4 6

Задачи по программированию

1. Сформировать матрицу размером 10x10, значение элементов которой равно произведению индексов соответствующего элемента.
2. Найти разность между максимальным и минимальным элементами массива X(15).
3. Вычислить сумму отрицательных элементов массива A(20), если их более 3, или найти произведения положительных элементов.
4. Заменить положительные элементы массива C(4, 3) нулями, а отрицательные их абсолютными величинами.

5. Если положительных элементов вектора K(24) больше, чем отрицательных, то отсортированные положительные элементы расположить в начале вектора. Иначе в начале вектора расположить отсортированные отрицательные элементы.
6. Первые пять элементов массива T(23) оставить в неизменном порядке, следующие 10 отсортировать в порядке возрастания, за ним три элемента оставить без изменения, а оставшиеся также отсортировать в порядке возрастания.
7. В каждом из нечетных столбцов матрицы A(4, 8) посчитать среднее арифметическое положительных элементов.
8. В каждой третьей строке матрицы X(9, 4) найти максимальный элемент и посчитать произведение найденных величин.
9. В матрице K(6,4) заменить отрицательные элементы единицами, а положительные – нулями. Вывести на экран преобразованную матрицу в общепринятом виде.
10. Отсортировать элементы вектора C(25), расположенные между максимальным и минимальным в порядке возрастания.

УКАЗАТЕЛИ

Указатель является переменной, которая содержит адрес другой переменной или функции. Суть концепции указателей состоит в работе с адресом ячейки памяти при косвенном доступе к ее содержимому. Описание указателя определяет тип данных, на которые указатель ссылается. Синтаксис объявления указателя:

<тип указываемых данных> *<имя указателя>;

Например:

```
int *int_ptr;          // указатель на целое
double *d_ptr;         // указатель на тип double
char *c;               // указатель на символьную переменную
int *array[10];        // объявление массива указателей, каждый из которых
                       // указывает на значение int
int (*pointer)[10];    // объявлен указатель с именем pointer, который указывает на
                       // массив из 10 элементов.
```

Можно использовать указатель на тип void, который способен указывать на объект любого типа. Его обычно называют пустым указателем. При выполнении операций над пустым указателем либо над объектом, на который он указывает, необходимо явно привести тип указателя к типу, отличному от void.

Указатели можно инициализировать при их объявлении, как и любые другие переменные. Например:

```
int result;
int *p_result=&result
```

Указатели и массивы логически связаны друг с другом. Имя массива является константой, содержащей адрес первого элемента массива, вследствие этого, значение имени массива не может быть изменено оператором присваива-

ния или каким-либо другим оператором. Так, если объявлен массив float temp[10]; то temp= =&temp[0].

Используя операции сложения, вычитания инкремента и декремента к указателю (или имени массива), можно передвигаться по элементам массива; temp++ будет указывать на второй элемент массива, т.е. элемент с индексом 1. Таким образом, можно работать с массивами, не используя их стандартного обращения к его элементам.

Пример 1

```
// программа демонстрирует работу с символьным массивом посредством
// указателей
#include <stdio.h>
#define SIZE 10
void main ( )
{
 char string[SIZE], *pc;
 int count;
 pc = string;          //в указатель помещается адрес первой ячейки массива
 printf("Введите %d символов", &SIZE);
 for (count =0; count<SIZE; count ++)
        {*pc=getchar( );    //занесение очередного символа по адресу
          pc++;
        }
//вывод на экран элементов массива в обратном порядке
pc= string+(SIZE-1);
 for (count =0; count<SIZE; count ++)
        {putchar(*pc);
         pc--;
        }
}
```

Необходимо соблюдать осторожность при использовании операций инкремента и декремента с указателями. Два следующих выражения будут иметь разный результат:

```
*pc++=getchar( );
*++pc=getchar( );
```

Первое выражение присваивает символ, возвращаемый функцией getchar( ), текущей ячейке, адресуемой указателем pc, после чего выполняется приращение самого указателя. Во втором выражении сначала происходит приращение указателя pc, после чего в ячейку по обновленному адресу будет записан результат функции getchar( ).

Указатели можно использовать и для многомерных массивов. Предположим, есть описание:

```
int zippo [4][2];
```

Поскольку первое число в объявлении означает количество строк, а второе – количество столбцов, то zippo = = &zippo[0][0], а zippo+5 указывает на элемент zippo[2][1].

Передача массива в функцию выполняется следующим механизмом: указываются тип элементов массива, его имя, а после пустые квадратные скобки, которые свидетельствуют, что данный аргумент является массивом. При вызове функции фактическим параметром функции является имя массива. Например:

int sum (int n, int array[]) // объявление функции
int b=sum(10, array); // вызов функции

и тогда, в действительности, функция получает адрес первого элемента массива, и, следовательно, она может быть вызвана следующим образом:

int c=sum(15, &arr[0]); где int arr[15];

Двумерный массив является одномерным массивом, элементы которого – одномерные массивы, следовательно, имя первой строки zippo[0], а имя четвертой строки – zippo[3]. Однако имя массива является также указателем на этот массив в том смысле, что ссылается на его первый элемент. Следовательно, zippo[0] = = &zippo[0][0], а zippo[2] = = &zippo[2][0]. Это позволяет использовать функцию, предназначенную для работы одномерного массива, для работы с двумерным массивом.

Пример 2

```
// функция нахождения среднего арифметического массива целых чисел
#include <stdio.h>
float mean(int array[], int n)
{ int index;
  long sum;
  if (n>0) { for (index=0,sum=0; index<n;index++)
             sum+=*(array+index);
             return(float)(sum/n);
           }
 else {cout<<”Нет массива”<<endl;
        return 0;
       }
}

void main ( )
{
 int matrix{ {2,4,6,9}, {10,20, 40,10},{3,7,0,9}};
 int line;
 for (line=0; line<3; line++)
 printf(“Средн. арифметич. %d строки равно %f \n”,line+1,mean (matrix [line], 4);
}
```

Пример 3

```
// Функция, вычисляющая сумму произведений первого элемента одномерного
// массива с последним, второго – с предпоследним и т. д.
// При работе с элементами массива используются указатели.
// Функция используется для каждой строки матрицы T (5, 6).
#include<stdio.h>
#include<conio.h>
int FUN(int*);       // объявление прототипа функции
void main( )
{
  int T[5][6]={{2,3,1,5,4,8},{6,7,4,2,3,9},{12,5,3,6,8,2},{3,4,5,9,4,2},{2,7,3,4,9,1}};
  clrscr( );              // очистка экрана
// вывод матрицы на экран в общепринятом виде
 for (int i=0; i<5; i++)
         {printf("\n");
           for (int g=0;g<6;g++)   cprintf(" %2d ",t[i][g]);
         }
 for (int i=0;i<5;i++)
 cprintf(" искомая сумма %d строки равна %3d\n",i+1,FUN(T[i], 6));
}
int FUN(int *massiv, int n)                //описание функции
{
 int sum=0;
  for(int m=0; m<n/2; m++)
  sum+=*(massiv+m)*massiv[n-m-1];
  return(sum);
}
```

Быстрая самопроверка

1. В приведенном фрагменте программы, возможно, содержится ошибка. Выбрать среди приведенных ответов правильный.

```
int arr[10] = {0, 1, 2, 3, 4, 5};
int i, s = 0;
for (i = 0; i < 5; ++i)
    s += *arr++;
```

a) всё правильно
b) в операторе присваивания допущена ошибка. Следует писать s += *(arr+i);
c) в операторе присваивания допущена ошибка. Следует писать s += arr++;

2. Если считать, что указатель занимает 2 байта, а символ 1 байт, то в процессе работы фрагмента программы на экран выведется:

```
char arr1[2][10] = {"Hello", "Bye"};
char *arr2[2] = {"Hello", "Bye" };
printf("%d %d", sizeof(arr1), sizeof(arr2));
```

a) 10 10 b) 20 8 c) 20 4 d) 8 8 e) 20 2 f) 20 20

3. После выполнения фрагмента программы:

```
float bin[4] = {0, 1, 0, 1};
int i, num0 = 0, num1 = 0;
for (i = 0; i <= 3; ++i)
switch (bin[i]) {
  case 0: num0++; break;
  case 1: num1++;
}
printf("Нулей %d, единиц %d", num0, num1);
```

на экран выведется

a) нулей 0, единиц 0 b) нулей 2, единиц 2 c) нулей 2, единиц 1
d) нулей 1, единиц 2 e) фрагмент работать не будет, т.к. в нем ошибка

4. int arr[5] = {0, 1, 2, 3, 4, 5};
В приведенном фрагменте программы, возможно, содержится ошибка. Выбрать правильный вариант ответа:
a) строка записана верно
b) строка записана неверно, т.к. число инициализирующих переменных больше размера массива
c) следует писать int *arr = {0, 1, 2, 3, 4, 5};

5. Присвоить значение 27.3 элементу, расположенному на 13 строке в 7 столбце массива float plan [30][20];
a) plan [13][7]=27.3; b)plan[7][13]=27.3;
c)plan[12][6]=27.3; d)plan[6][12]=27.3

6. Функция с именем prim возвращает указатель на символ и имеет два аргумента целого типа. Какой из вариантов прототипа функции является правильным:
a) prim (char* c, int x, int y); b) void prim (char*c, int x, int y);
c)*char prim (int x, int y); d) char* prim(int x; int y);

Задачи по программированию

1. Подсчитать количество элементов матриц X(10, 3) и Y(6, 4), удовлетворяющих условиям $0 \le x_{ij} \le 2,\ 3 \le y_{ij} \le 5$.
2. Вычислить суммы положительных элементов каждой строки для матриц A(3, 8) и B(2, 9).

3. Вычислить значение $z = ((a,d) - (a,b))/(c,c)$, где *a, b, c, d* – векторы размерности 10, причем *c* – вектор, содержащий наименьший максимальный элемент, *(a, b)* – скалярное произведение векторов *a* и *b*, т.е $(a, b) = \sum_{i=1}^{10} a_i b_i$.
4. Создать функцию, вычисляющую сумму значений, расположенных между максимальным и минимальным элементами массива. При работе с элементами массива использовать указатели. Воспользоваться функцией для каждой строки матрицы O(3, 8).
5. Создать функцию, записывающую элементы массива в обратном порядке. При работе с элементами массива использовать указатели. Воспользоваться функцией для каждой строки матрицы K (5, 5).
6. Создать функцию, параметром которой будет одномерный массив. Посчитать среднее арифметическое элементов, стоящих на нечетных местах. При работе с элементами массива использовать указатели. Воспользоваться функцией для каждой строки матрицы M (4, 6).
7. Вычислить значение

$$y = \begin{cases} \sum_{i=1}^{10} a_i^{\,2}, & \text{если } \sum_{i=1}^{10} a_i b_i < 0, \\ \sum_{i=15}^{20} a_i b_i \Big/ \sum_{i=10}^{15} b_i^{\,2}, & \text{иначе} \end{cases}$$

где *a, b* – векторы размерности 20. Использовать функцию для нахождения суммы.
8. Отсортировать элементы массива P(30), расположенные между максимальным и минимальным элементами, в порядке возрастания.
9. В каждом столбце матрицы, номер которого кратен 3, вычислить разность между суммами положительных и отрицательных элементов.
10. Определить, какой из векторов: A(10), B(15),C(20) – содержит наибольшее количество неотрицательных элементов.

ССЫЛКИ

Ссылка – указатель, который не требует разыменовывания при использовании. Разница между указателем и ссылкой заключается в том, что программист может использовать ссылку как обычный объект, несмотря на то, что к объекту будет производиться косвенный доступ, в то время как указателю необходимо явно присвоить значение адреса объекта.

Например:

```
int i = 123;
int *p=&i;  // &i является ссылкой на объект с именем i
```

Ссылки используются в качестве переменных, параметров и результатов функций. Необходимо помнить, что однажды инициализировав ссылку, ей уже нельзя присвоить другое значение.

Ссылки получили широкое применение в качестве параметров функций, особенно полезны они в функциях, которые возвращают несколько значений.

При этом функция не возвращает значение напрямую. Обращения к переменным из вызывающей функции происходит по их адресам.

Пример 1

```
// Функция, производящая обмен значениями двух переменных
#include <iostream.h>
void exchange(int &a, int&b) // параметры функции – ссылки
{ int c=a;
  a=b;
  b=c;
}
void main( )
{
 int a=100, b=10;
 cout<<”До обмена : a=” <<a<<”,b=”<<b<<endl;
exchange (a,b);
cout<<” После обмена: a=” <<a<<”,b=”<<b<<endl;
}
```

Доступ к переменной осуществляется как через ее имя, так и через имя-синоним (ссылку) в вызываемой программе. После завершения вызываемой программы (функции) имя-синоним уничтожается, однако измененное значение переменной в вызываемой функции сохраняется.

Пример 2

```
// В программе описана функция, получающая в качестве аргументов
// некоторый вес, выраженный в килограммах и граммах, а также величину
// увеличения данного веса в граммах. Функция возвращает новый вес,
// выраженный в граммах и килограммах.
#include<iostream.h>
#include<conio.h>
void massa(int& kg, int& gr, int dgr);
void main( )
{
        clrscr();
        cout<<"Введите вес: килограммы, граммы \n";
        int kg, gr;
        cin>>kg>>gr;
        cout<<"Введите прибавку в весе в граммах \n";
        int dgr;
        cin>>dgr;
        massa(kg, gr, dgr);
        cout<<"Новый вес –  "<<kg<<"кг "<<gr<<"гр";
        getch();
}
void massa(int& kg,int& gr,int dgr)
```

```
{       gr+=dgr;
        while(gr>=1000){ gr –=1000;
                         kg++;
                        }
}
```

Задачи по программированию

1. Найти наименьшие элементы и номера строк и столбцов, в которых они расположены, для матриц A(4, 7) и B(9, 6).
2. Вычислить сумму и количество элементов матриц K(7, 7) и H(4, 4), расположенных над главной диагональю.
3. Вычислить $z = \frac{s_1 + s_2}{k_1 k_2}$, где s_1, k_1 – сумма и количество положительных элементов массива A(10), s_2, k_2 – сумма и количество положительных элементов массива B(15).
4. Вычислить сумму и количество отрицательных элементов матриц A(5, 5) и C(6, 6), расположенных под главной диагональю и на ней.
5. Для каждого из 4 цилиндров с известными радиусами основания и высотой вычислить площадь поверхности и объем. Площадь поверхности цилиндра и его объем находить в одной функции.
6. Создать функцию нахождения модуля и аргумента для комплексного числа по известным значениям вещественной мнимой частей.
7. Создать функцию, выполняющую преобразование декартовых координат в полярные.
8. Создать функцию, вычисляющую среднее арифметическое отрицательных элементов, для каждого десятка элементов вектора M(35).
9. Посчитать количество гласных и согласных букв символьного массива.
10. Создать функцию, аргументами которой являются три вещественных числа, f, b, g, меняющую их по следующему правилу: $f = \sqrt{\max(f, b, g)}$, $b = (\min(f, b, g))^3$, $g = f + b + g$.

СТРУКТУРЫ И ОБЪЕДИНЕНИЯ

Структуры позволяют определять новые типы данных путем логического группирования переменных различных типов.

Описание структурного шаблона выглядит следующим образом:

```
struct <имя шаблона> { <тип и имя поля структуры>;
                       ……
                       <тип и имя поля структуры>;
                      };
```

Например:

```
struct men
       {
```

```
  char fam[15];
  char name[15];
  int telefon;
  char manth[10];
  int day;
} ;
```

Объявление структурной переменной аналогично объявлению обычной переменной, но перед именем типа указывается ключевое слово struct, например:

```
struct men child;                  // описание структурной переменной
struct men men1,men2,*ptm;//описание двух структурных переменных и
                                   //указателя на структурную переменную
struct men peopl [100];            //объявлен массив из 10 элементов типа book
```

Имя типа структуры можно опускать в случае, если структурный шаблон используется один раз. Тогда происходит объединение в один этап определения структурного шаблона и структурной переменной. Например:

```
struct {float x, y; }complex;
```

Инициализация может быть сделана следующим образом:

```
struct men women={“Иванова”, “Ольга”, 360867, “январь”, 19};
```

Доступ к элементам поля структуры осуществляется через точку, например:

```
scanf ( “%d”,&women.telefon);
printf( “%s”,mens[0].name);
```

Объединение – средство, позволяющее запоминать данные различных типов в одном и том же месте памяти. В каждый момент времени объединение может хранить значение только одного типа из набора. Память, которая выделяется переменной типа объединение, определяется размером наиболее длинного из элементов объединения. Все элементы объединения размещаются в одной и той же области памяти, с одного и того же адреса. Значение текущего элемента теряется, когда другому элементу объединения присваивается значение.

Синтаксис определения объединения аналогичен определению структуры, с использованием ключевого слова union.

```
union sign { int svar;
             unsigned uvar;
           } number;          // Знаковое или беззнаковое целое
```

Возможно объявление указателей на структуру или объединение, и тогда для получения доступа к отдельным элементам структуры применяется указатель и оператор –>.

Пример 1

```
//Программа демонстрирует механизм объявления и инициализации
//структурной переменной с использованием указателей
```

```
#include <stdio.h>
struct boad{                              //объявление структурного шаблона
            char model[20];
            char nomer[8];
            int year;
            float price;
          };
int main (void)
{
int i, number;
struct boat boats[30], *ptr_boat;         // объявление массива структур и указателя
                                          // на структуру
ptr_boat=&boats[0];                       // инициализация указателя
printf («информацию о скольких лодках будем вводить?»);
scanf (“%d”,&number) ;
for (i=0; i<number; i++)
        {
        printf (“введите модель судна”);
        gets(ptr_boat–>model);                                  // ввод строки
        printf (“введите регистрационный номер судна”);
        gets(ptr_boat–>nomer);
        printf (“введите модель судна”);
        scanf(“%d”,&ptr_boat–>year);
        printf (“введите стоимость судна”);
        scanf(“%f”,&ptr_boat–>price);
        ptr_boat++;                        // переход к следующему элементу массива
        }
        …
        return (0);
}
```

Пример 2

```
// Программа, выдающая по запросу список книг в алфавитном порядке с
// указанием фамилии автора, число книг издания необходимого года, и список
// книг конкретного автора.
// Для организации данных используется динамический массив структур
#include<iostream.h>
#include<iomanip.h>
#include<string.h>
#include<conio.h>
void main( )
{
textcolor(15);
textbackground(1);
```

```
struct library
        { int shifr;
          char author[15];
          char title[15];
          int year;
        };
int i,n,m,god,v=0,g=0;
struct library *ptr;
ptr=new library[25];
if(ptr==NULL) cout<<"память не выделена";
char avtor[15];
cout<<"Введите количество книг\n";
cin>>n;
for(i=0;i<n;i++)
  { cout<<"Введите сведения о "<<(i+1)<<" книге :\n";
    cout<<"Шифр книги\n";
    cin>>ptr[i].shifr;
    cout<<" Фамилия автора\n";
    cin>>ptr[i].author;
    cout<<"Название\n";
    cin>>ptr[i].title;
    cout<<"Год издания\n";
    cin>>ptr[i].year;
  };
clrscr();
do
{
cout<<"Выберите действие \n";
cout<<"1 – список книг в алфавитном порядке\n";
cout<<"2 – список книг необходимого года\n";
cout<<"3 – список книг определенного автора\n";
cout<<"4 – выход\n";
cin>>m;
switch(m)
{
case 1:
                {
cout<<"*****************************************************\n";
cout<<"* шифр книги *      автор      *    название        *   год издания  *\n";
cout<<"*****************************************************\n";
          for(char w='A';w<'Z';w++)
          for(i=0;i<n;i++ )
          if(ptr->title[0]==w)
```

```
        cout<<"*"<<setw(5)<<ptr[i].shifr<<"  *"<<setw(15)<<ptr[i].author<<"
*"<<setw(15)<<ptr[i].title<<"  *"<<setw(10)<<ptr[i].year<<"   *\n";
cout<<"*************************************************\n";
        cout<<"Для выхода в меню нажмите любую клавишу";
        getch();
        clrscr();
        break;
              };
case 2:
              {
        cout<<"Введите нужный год\n";
        cin>>god;
        for(i=0;i<n;i++)
              {
              if(ptr[i].year==god)
              {
                    v+=1;
                    if(v==1)
              {
      cout<<"*************************************************\n";
      cout<<"* шифр книги *     автор     *    название        *   год издания  *\n";
      cout<<"*************************************************\n";
                    };
cout<<"*"<<setw(5)<<ptr[i].shifr<<"  *"<<setw(15)<<ptr[i].author<<"
*"<<setw(15)<<ptr[i].title<<"  *"<<setw(10)<<ptr[i].year<<"   *\n";
              };
              };
                    if(v!=0)
      cout<<"*************************************************\n";
              else cout<<"Нет книг данного года издания\n";
                v=0;
             cout<<"Для выхода в меню нажмите любую клавишу";
        getch();
        clrscr();
              break;
      };
case 3:
      {
        cout<<"Введите фамилию автора \n";
        cin>>avtor;
        for(i=0;i<n;i++)
              {
              if(strcmp(ptr[i].author,avtor)==0)
              {
                    g+=1;
```

```
                    if(g==1)
                  {
cout<<"*****************************************************\n";
cout<<"* шифр книги *     автор     *    название        *   год издания  *\n";
cout<<"*****************************************************\n";
                  };
  cout<<"*"<<setw(5)<<ptr[i].shifr<<"  *"<<setw(15)<<ptr[i].author<<"
*"<<setw(15)<<ptr[i].title<<"  *"<<setw(10)<<ptr[i].year<<"   *\n";
                 };
                 };
                 if(g!=0)
cout<<"*****************************************************\n";
                 else  cout<<"Нет книг данного автора\n";
                   g=0;
                cout<<"Для выхода в меню нажмите любую клавишу";
          getch();
          clrscr();
                 break;
        };
case 4:
         { delete [] ptr;       // удаление массива из динамической памяти
           break;
         };
        };
}
while(m!=4);
clrscr();
}
```

Быстрая самопроверка

1. Даны объявления:

```
typedef char Code[26];
enum Style {Formal, Brief}          //перечисляемый тип
struct Ref
    {Code token [2000], symbol [20];}
struct MapType
    {
     Code mapCode;
     Style style1;
     Ref chart;
    };
 MapType guide[200], aMap;
 Ref aRef;
 int count;
 Code aCode;
```

Отметьте каждое из перечисленных ниже выражений как либо правильное, либо неправильное.

a) if (aMap.style1 ==Brief) count++;
b) guide[1].chart.token[2]=aMap;
c) guide[6].chart=aRef;
d) guide[100].chart.token[1][2]=aCode[2];
e) guide[20].token[1]=aCode;
f) if (guide[20].style1 == Formal) guide[20].chart.token[0] [0]='A';
g) aMap=guide[5];
h) aMap.chart=aRef;

Задачи по программированию

1. Компьютер может поддерживать до 30 терминалов. Для каждого из них должна храниться следующая информация:
   название и модель;
   скорость передачи данных;
   дуплексность (Полудуплекс, Полный дуплекс);
   биты данных (целое число со значением 7 или 8);
   стоповые биты (целое число со значением 1 или 2).
   Разработать структуру данных для учета терминалов. Написать объявление созданных переменных.
2. Написать объявление структуры для хранения строки не более чем из 20 символов и длины строки. Затем создать функцию, возвращающую длину строки, содержащейся в этой записи.
3. Написать программу для управления складом компании. Для каждого товара должна храниться следующая информация:
   номер товара;
   название товара;
   цена;
   количество товара.
   Организовать выдачу информации о товаре на складе по неполным данным.
4. Создать компьютерную записную книжку, которая должна содержать следующие сведения:
   фамилия,
   имя,
   дата рождения,
   телефон.
   По требованию выдавать список фамилий в алфавитном порядке с указанием номеров телефонов. По запросу пользователя должен выводиться на печать список тех, кого необходимо поздравить с днем рождения в заданном месяце (с указанием числа).
5. Создать два массива, содержащие сведения о пяти нападающих хоккейных команд «Спартак» и «Динамо»:

фамилии нападающих;
число заброшенных ими шайб;
число, сделанных голевых передач;
штрафное время.
Написать программу, которая по данным из этих массивов создает массив, содержащий
фамилию;
команду;
сумму очков (голы + передачи)
для шести лучших игроков обеих команд и выводит его содержимое на экран.

6. Создать массив, содержащий данные о пациентах скорой помощи:
фамилия,
пол,
год рождения,
диагноз,
информацию о госпитализации (да, нет).
Создать программу, выводящую на экран в алфавитном порядке госпитализированных больных с указанием диагноза, а также список детей с заболеванием X (X – вводится с клавиатуры) с указанием возраста и пола.
6. Багаж пассажира характеризуется количеством вещей и указанием веса каждой вещи. Вывести на экран общий вес багажа каждого пассажира, найти багаж, средний вес одной вещи в котором отличается не более чем на 0,3 кг от общего среднего веса вещей, определить количество пассажиров, имеющих более трех вещей в багаже.
7. Создать шаблон для хранения информации для пейджинговой связи. Переменная должна содержать информацию о состояния пейджера: доступен или нет, а также в случае получения информации – ее содержание.

ФАЙЛЫ

В C++ существует два вида файлов – текстовые и двоичные (бинарные). Текстовым считается файл, в котором информация запоминается в виде символов кода ASCII (или аналогичном). Он отличается от бинарного файла, который обычно используется для запоминания кодов машинного языка.

Библиотека C++ содержит три класса, с помощью которых можно управлять файловым вводом-выводом:

ifstream подключает к программе файл, предназначенный для ввода данных (входной файловый поток),

ofstream подключает к программе файл, предназначенный для вывода данных (выходной файловый поток),

fstream подключает к программе файл, предназначенный как для ввода, так и для вывода.

Для подключения данных классов необходимо подключить файл ifstream.h. Чтобы создать объект класса ifstream и связать с ним файл, находящийся в текущем каталоге, необходимо записать следующее:
ifstream ifsin (“text.txt”, ios::in);

Итак, создается объект с именем ifsin класса ifstream и связывает с ним файл text.txt. Если файл, с которым связывается объект, находится не в текущем каталоге, необходимо полностью указывать путь. Во втором аргументе указывается один из следующих флагов:

Флаг	Назначение
ios::in	Файл открывается для чтения, его содержимое не открывается
ios::out	Файл открывается для записи
ios::ate	После создания объекта маркер текущей позиции устанавливается в конец файла
ios::app	Все выводимые данные добавляются в конец файла
ios::trunc	Если файл существует, его содержимое очищается автоматически

Пример 1

```
// В файле содержится некоторое количество чисел. Сформировать из них
// матрицу, содержащую 4 столбца. Недостающие элементы  последней строки
// обнулить. Вывести на экран матрицу в общепринятом виде, ее размерность и
// сумму элементов главной диагонали
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<string.h>
void fun(char *string);
void main()
{
int ss;
clrscr();
printf("Выберите действие:\n");
printf("1- Использовать созданный файл \n");
printf("2- Создать новый файл \n");
scanf("%d",&ss);
if(ss= =1)
{
    fun("a:/text1.txt");
          getch();
};
if(ss= =2)
      {   fstream in("a:/text2.txt",ios::out);
            char dh;
            scanf("\n");
            do
```

```
            {
            scanf("%c",&dh);
            if(dh!='\n') in<<dh;
             }
             while(dh!='\n');
             in.close();
       fun("a:/text2.txt");
       getch();
            }
}

void fun(char *string)
{
int i,j,x=0,sum=0,matr[20][4],c;
long nn;
fstream inn(string,ios::in);
  if(!inn) cerr<<" ????«\n";
  else
  {
  printf("б®¤Ґа¦Ё¬®Ґ д ©« :\n");
  while(nn!=EOF)
     {
      for(i=0;i<4;i++)
            {
              inn>>c;
              matr[x][i]=c;
              nn=inn.get();
              printf("%d ",c);
              if(nn==' ')nn=inn.get();
              if((nn==EOF)&&(i<3)){
                            for(int g=i+1;g<4;g++)
                            matr[x][g]=0;
                            break;
                           };
             if(nn!=EOF) { int e=inn.tellp();
                            inn.seekg(e−1);
                     };
            };
          if(nn==EOF) break;
          x++;
      };
     printf("Матрица:\n");
     for(i=0;i<x+1;i++)
          {
             for(j=0;j<4;j++)
```

```
                {
                 printf("%d\t",matr[i][j]);
                 if(i==j)  sum+=matr[i][j];
                };
            printf("\n");
        };
        printf("Число строк матрицы: %d:4\n",x+1);
        printf("сумма элементов главной диагонали : %d", sum);
        }
return;
}
```

С++ также поддерживает приемы работы с файловыми переменными, широко применяемыми еще в языке С.

Для работы с файлом описываем указатель на переменную типа FILE:

FILE *in;

Открытие файла происходит с помощью функции fopen( ), которой управляют три основных параметра:

имя файла, который следует открыть, – оно является первым аргументом;

второй аргумент указывает на то, как будет использоваться файл: “r” – для чтения, “a” – для дополнения, “w” – для записи, в случае бинарного файла к каждому из аргументов добавляется “b”;

указатель на файл – это значение возвращается функцией.

Функция закрытия файла fclose( ) имеет только один аргумент – указатель на файл.

Ввод-вывод текстового файла осуществляется с помощью функций getch( ) и putch( ), которые соответственно получает и записывает символ.

Пример 2

```
//В данном текстовом файле с именем text.txt посчитать количество слов, длина
//которых меньше семи.
#include<stdio.h>
#include<iostream.h>
 void main( )
{file*f;  char ch;
  int count=0, sum=0;
  if (f=fopen(“text1.txt)!=NULL)
      do {     ch=getch(f);
               if (ch!=’ ‘)  count++;
               else  if (count<7)  {sum++; count=0; }
          }
      while (ch!=eof(f));
      cout<< “В файле содержится”<<sum<<”слов, длина которых меньше семи”;
  }
```

Для работы с бинарными файлами существует ряд функций:

FREAD (buffer, size, count, stream) – всего 4 аргумента. Данная функция читает count элементов длины size из входного потока (файла) stream (FILE * stream) и помещает в заданный массив buffer. При этом указатель файла увеличивается на число действительно прочитанных байтов.

FWRITE имеет те же аргументы. Функция дописывает count записей, по size байтов каждый, из области buffer в выходной поток stream.

FSEEK (stream, offset, origin) перемещает внутренний указатель файла, связанный с потоком stream, на новое место в файле, которое вычисляется по смещению offset и указанию направления отсчета origin. Следующая операция ввода/вывода с указанным потоком stream будет выполнена, начиная с той позиции, на которую произведено перемещение. Аргумент offset должен быть типа long, а origin должен принимать значение одной из следующих целочисленных констант:

SEEK_SET (значение 0) – начало файла,

SEEK_CUR (значение 1) – текущая позиция указателя файла,

SEEK_END (значение 2) – конец файла.

Пример 3

```
//В файле содержатся числа, сформировать из них матрицу 4X4, взяв первые 8
//чисел сначала файла, а остальные – с конца файла. Посчитать произведение
//элементов под главной диагональю матрицы
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
void main( )
{
        int i, j, n, proisv=1, rasmer;
        int array[50], buf[4][4];
        FILE *stan;
        long int pos;
        fpos_t filepos=0, filepos_1=0;
        stan=fopen("anton.dat","rb");    //открытие бинарного файла для чтения
        for(i=0;i<2;i++)
        for(n=0;n<4;n++)
        buf[i][n]=getc(stan);
        fseek(stan, EOF, SEEK_END);     //переход на последний элемент файла
        for(j=2;j<4;j++)                 //поэлементное чтение элементов с конца
        for(n=0;n<4;n++)                 //файла в матрицу
                {       if(j==2)
                        {       fseek(stan,EOF-(n),SEEK_END);
                                buf[j][n]=getc(stan);
                        }
                        else
                        {       fseek(stan,EOF-(n+4),SEEK_END);
                                buf[j][n]=getc(stan);
```

```
                }
            }
            fclose(stan);
        clrscr();
            for(i=0;i<4;i++)
            {
                for(n=0;n<4;n++)
                printf("%3d ",buf[i][n]);
                cout<<"\n";
            }
                        for(i=0; i<4; i++)
            for(j=0; j<4; j++)
                if (j<i) proisv*=buf[i][j];
            cout<<"Произведение элементов под главной диагональю =";
            cout<<proisv;
        }
   else cout<<"error!\n";
}
```

Быстрая самопроверка

1. Исправить следующую программу таким образом, чтобы она читала данные из файла inData, а сохраняла результаты в файле outData.

```
#include <iostream.h>
void main ( )
{
int n;
ifstream inData;
outData.open (“result.dat”);
cin>>n;
outData<<n<<endl;
}
```

2. Использовать исправленный вариант программы из предыдущего задания для ответов на следующие вопросы:
   a) если файл inData первоначально содержит значение 144, что он будет содержать после выполнения программы?
   b) если файл outData перед запуском программы пуст, что будет в нем содержаться после окончания работы программы?

Задачи по программированию

1. В файле содержится некоторое количество чисел. Сформировать из них матрицу, содержащую три столбца. Лишние элементы в неполной строке отбросить. Вывести на экран матрицу в общепринятом виде, ее размерность и сумму элементов побочной диагонали.

2. В файле содержатся числа. Сформировать из них матрицу, содержащую пять элементов в строке. Лишние числа отбросить. Вывести на печать матрицу в общепринятом виде и посчитать сумму элементов, содержащихся в предпоследней строке.
3. Даны два текстовых файла. Слить содержимое файлов, чередуя во вновь созданном файле слова из разных файлов.
4. В текстовом файле выбрать слова, состоящие из семи букв, и вывести их на печать в алфавитном порядке.
5. В файле содержатся числа. Сформировать квадратную матрицу, содержащую последние 16 чисел файла. Посчитать сумму элементов над главной диагональю полученной матрицы.
6. В текстовом файле найти все симметричные слова и вывести их на экран.
7. Из цифр, содержащихся в файле, сформировать новый файл следующим образом: сначала расположить трехзначные цифры, затем двузначные. Найти максимальное и минимальное число в полученном наборе.
8. В файле задан текст. Выбрать из него второе и четвертое предложения. Посчитать количество букв в каждом из предложений и вывести на экран меньшее из них по длине.
9. В файле задан текст. Изменить его, заменив все символы ‘а’ на символ ‘о’ в третьем с конца предложении и символы ‘и’ на символ ‘е’ во втором предложении, считая с начала файла.
10. Посчитать количество слов в файле, начинающихся с введенной буквы. Вывести на печать максимальное из них.

КЛАССЫ

Класс есть расширение понятие структуры языка C++. Он позволяет создать типы и определять функции, которые задают поведение типа. Каждый представитель класса называется объектом.

Объединение данных и функций, которые обрабатывают объект определенного типа, называется инкапсуляцией.

Ключевые слова – **Private** (закрытый), **Public** (открытый), **Protected** (защищенный) – спецификаторы доступа. По умолчанию элементы класса являются закрытыми.

Открытые данные-члены и члены-функции доступны снаружи класса. Они отвечают за внешний интерфейс класса.

Закрытые данные-члены и члены-функции предназначены для внутреннего использования в классе. Их могут использовать только функции-члены класса.

Защищенные элементы данных и члены-функции доступны для функций-членов данного класса и классов, производных от него.

C++ имеет две встроенные особенности – конструкторы и деструкторы, помогающие не забывать про инициализацию переменных-членов класса и про уничтожение ненужных объектов.

Конструкторы – это процедуры, которые автоматически вызываются при создании объекта (инициализации класса).

Можно поместить процедуры инициализации внутрь конструкторов для того, чтобы произвести нужные установки в начале использования объекта. Класс может содержать несколько конструкторов, чтобы по-разному инициализировать классы.

Когда закончена работа с объектом, то автоматически вызывается функция, называемая деструктором.

Конструктор и деструктор носят имя класса, членами которого они являются, только перед именем деструктора ставится тильда (~).

Пример 1

```
# include<iostream.h>
class Matrix
{ int i, j; float m[3][3], k;
  public:       Matrix (float a[3][3], float c=1); //объявление конструктора
                Void out();          //объявление функции –элемента класса
};
Matrix :: Matrix (float a[3][3], float c)   //определение конструктора
{k=c;
for (i=0; i<3; i++)
for (j=0; j<3; j++)
m[i][j]=a[i][j];}
void  Matrix::out( )
{ for (i=0; i<3; i++)
{ for (j=0; j<3; j++)
cout <<(m[i][j]+k)<<;
cout <<endl;}
}
void main( )
{ float M[3][3]={{1,2,3}, {4,5,6}, {7,8,9}};
Matrix first(M);
first out( );
Matrix second (M, 5);
Second out( );
}
```

Наследование – это механизм, позволяющий строить иерархию классов. Простое наследование описывает родство между двумя классами. Класс, являющийся прародителем, называется базовым классом, прочие – производными классами. Из одного базового класса могут выводиться многие классы. Производный класс сам может быть базовым, который наследуют другие классы. Производный класс наследует из базового класса данные-функции и данные-члены, деструкторы, но не конструкторы.

Пример 2

```
//Организован класс прямоугольник, содержащий конструктор, деструктор,
//функцию нахождения площади. Создан производный класс пирамида,
//определенный основанием и высотой и содержащий функции нахождения
//площади поверхности и объема пирамиды.
#include<stdio.h>
#include<iostream.h>
#include<conio.h>
#include<math.H>
void main()
{
class pryamoug                          // базовый класс
{protected: int x,y;
 public:
 pryamoug( ){};
 pryamoug(int _x,int _y){x=_x; y=_y;};  //конструктор
 ~pryamoug(){};                         //деструктор
 int S( ){ return (x*y); };
};
class piramida : public pryamoug        //производный класс
{int h,s;
public: piramida(int _x,int _y,int _h)  //конструктор
{x=_x; y=_y; h=_h; s=pryamoug::S();};
float pover( )                          // нахождение площади поверхности
  { float p,a;
   p=2*(x+y);
   a=sqrt(h*h+sqrt((x/2)*(x/2)+(y/2)*(y/2)));
   return (p*a/2);
  }
  float V( ) { return (s*h/3); }        //нахождение объема
};
int x1,y1,h1;
cout<<"Введите ширину основания";
cin>>x1;
cout<<"Введите длину основания";
cin>>y1;
cout<<"Введите высоту пирамиды";
cin>>h1;
primoug A(x1,y1);
piramida B(x1,y1,h1);
printf("Площадь основания пирамиды %4d\n",A.S());
printf("Площадь поверхности пирамиды %4.2f \n", B.pover() );
printf("Объем пирамиды %3.0f \n",B.V());
getch();
}
```

Существует также множественное наследование. Для вывода нового класса из нескольких базовых следует перечислить имена базовых классов после имени нового.

Пример 3

```
//Объявлен класс Позиция, определяющий координаты на экране.
//Объявлен класс Строка, содержащий конструктор и функцию вывода строки
//на экран.
//Объявлен производный класс  Позиция + Строка, содержащий функцию
//вывода строки с нужной позиции
#include <iostream.h>
#include <string.h>
#include <conio.h>
class position
{
protected:    int x,y;
public:       void setcoord(int _x,int _y)
              {x=_x;y=_y;};
};
class stroka
{
protected:    char str[80];
public:       stroka(char st[80]="stroka")
              {strcpy(str, st);};
              void draw()
              {cout<<str;};
};
class vivod : public position, public stroka
{
public:       vivod(char st[80]) : stroka(st){};
              void draw( )
              {gotoxy(x,y);cout<<str;};
};
void main( )
{
clrscr( );
int xx, yy;
char s[80];
cout<<"Введите значение координаты x:";  gotoxy(25,1); cin>>xx; gotoxy(1,2);
cout<<"Введите значение координаты y: ";  gotoxy(25,2); cin>>yy;
cout<<"Введите строку:";  cin>>s;
clrscr();
vivod a(s);
textbackground(WHITE);
textcolor(BLUE);
```

```
clrscr();
a.setcoord(xx,yy);
a.draw();
cout<<"("<<xx<<","<<yy<<")";
getch();
}
```

Быстрая самопроверка

1. В приведенном фрагменте программы, возможно, содержится ошибка. Выбрать среди приведенных ответов правильный.

```
class A {     int x;
                public: void SetX (int _x) { x = _x; }
          };
A *a;
SetX (10);
```

a) всё правильно;
b) для доступа следует писать A.SetX (10);
c) для доступа следует писать a.SetX (10);
d) для доступа следует писать A –> SetX (10);
e) для доступа следует писать a –> SetX (10);
f) среди приведённых ответов нет правильного.

2. Выбрать правильный вариант результата работы программы:

```
main ( )
{ class A { public:   A( ) { printf ("construct A\n");}
                        ~A( ) { printf ("destruct A\n");}
          };
class B : A { public:   B( ) { printf ("construct B\n");}
                ~B ( ) { printf ("destruct B\n");}
                };
B b;
}
```

a) ничего не выведется

b) фрагмент не будет скомпилирован

c)
```
construct A
destruct A
construct B
destruct B
```

d)
```
construct A
construct B
destruct B
destruct A
```

e)
```
construct B
destruct B
```

f)
```
construct B
construct A
destruct A
destruct B
```

3. Рассмотреть приведенное ниже объявление класса и код клиента:

Объявление класса

```
class SomeClass
{
public:
    void Func1(int n);
    void Func2( );
private:
    int SomeInt;
 }
```

Код клиента

```
SomeClass object1;
SomeClass object2;
int m;

object1.Fun1(3);
```

Перечислить все идентификаторы, являющиеся именами членов класса.
Перечислить все идентификаторы, являющиеся именами объектов класса.
При реализации класса SomeClass какое из следующих описаний функций будет корректным для Func2:
a) void Func2 ( ) { . . . }
b) void SomeClass : : Func2 { . . . }
c) SomeClass : : void Func2 { . . . }

Задачи по программированию

1. Организовать класс *Комплексное число*, содержащий конструктор, деструктор и функции нахождения модуля и аргумента комплексного числа, а также функцию вывода. Организовать производный класс, содержащий два комплексных числа и функции нахождения умножения, деления и сложения комплексных чисел. Продемонстрировать в программе работу всех функций.
2. Организовать класс *Матрица,* содержащий конструктор, деструктор, функцию вывода матрицы в общепринятом виде. Организовать производный класс, содержащий две матрицы и функции нахождения произведения и сложения матриц. Продемонстрировать в программе работу всех функций.
3. Организовать класс *Треугольник,* определенный по трем сторонам содержащий конструктор, деструктор, функцию нахождения периметра и площади (по формуле Герона). Организовать производный класс, содержащий дополнительно функцию нахождения углов треугольника, высоты и новой функции нахождения площади по основанию и высоте. Продемонстрировать в программе работу всех функций.
4. Организовать класс *Окружность,* определяемый координатой центра и радиуса. Заданный класс должен содержать конструктор, деструктор, функцию вычисления площади круга. Организовать производный класс *Конус*, определенный по радиусу основания и координатой вершины и содержащий функции нахождения площади поверхности и объема конуса. Продемонстрировать в программе работу всех функций.
5. Организовать класс *Дробь,* содержащий конструктор, деструктор, функцию вывода дроби в общепринятом виде и функцию выделения целой части. Организовать производный класс, содержащий две дроби и функции вычисле-

ния сложения и вычитания дробей. Продемонстрировать в программе работу всех функций.

6. Организовать класс *Дробь,* содержащий конструктор, деструктор, функцию вывода дроби в общепринятом виде и функцию выделения целой части. Организовать производный класс, содержащий две дроби и функции вычисления деления и умножения дробей, функцию приведения дроби к несократимому виду. Продемонстрировать в программе работу всех функций.
7. Описать класс *Дата,* содержащий данные – число, месяц, год. Описать конструктор и функцию, проверяющую правильность введенной даты. Описать производный класс, включив в дату день недели. Описать функцию, для определения дня недели, на который приходится введенная дата, если считать, что 1-е января 1-го года нашей эры – понедельник, функцию вывода даты на экран.
8. Описать класс *Карта* (масть и достоинство), содержащий конструктор и функцию вывода на экран. Описать производный класс, содержащий 2 карты и козырь, и функцию, проверяющую, кроет ли первая карта вторую с учетом козыря. Продемонстрировать работу функций.
9. Описать класс *Треугольник*, содержащий координаты вершин, конструктор, функцию, определяющую правильность введения данных, т.е. проверяющую возможность построения треугольника по заданным вершинам, и функцию, рисующую треугольник на экране. Описать производный класс *Треугольник + цвет,* дополнительно содержащий цвет и функцию, закрашивающую треугольник в заданный цвет. Продемонстрировать в программе работу всех функций.
10. Описать класс *Окно* (прямоугольная рамка), с функцией перемещения окна. Описать производный класс *Окно с заголовком,* содержащий конструктор и наследующий все свойства базового класса *Окно*. Продемонстрировать работу всех функций в программе.

ОТВЕТЫ НА ЗАДАНИЯ САМОПРОВЕРКИ

Функции

1. c
2. b
3. float function (int x, int k)
   {
   float z=5, k;
   z+=(x*x)/(2+k);
   return z;
   }

Массивы

1. e
2. b
3. c
4. a

Указатели

1.b
2. c
3. e
4. b
5. a
6. c

Структуры

1. a) правильно b) правильно c) неправильно d) правильно e) неправильно f) правильно g) правильно

Файлы

1. #include <iostream.h>
   void main ( )
   {
   int n;
   ifstream inData;
   ifstream outData
   outData.open (“result.dat”);
   outData>>n;
   outData<<n<<endl;
   }

   2. a) 144 b)144 после чего символ перевода строки

Классы

1.e
2.d

Библиографический список

1. *Дейл Н., Уимз Ч., Хедингтон М.* Программирование на C++. М.: ДМК, 2000.
2. *Паппас К., Мюррей У.* Программирование на C и C++. Киев: Изд. группа BHV, 2000.
3. *Скляров В.А.* Язык C++ и объектно-ориентированное программирование. Минск: Выш. шк., 1997.
4. *Козелл Е.И. и др.* От Си к C++. М.: Финансы и статистика, 1993.
5. *Бочков С.О., Субботин Д.М.* Язык программирования Си для персонального компьютера. М.: СП «Диалог»; «Радио и связь», 1990.

СОДЕРЖАНИЕ

Татьяна Алексеевна Галаган,
ст. преподаватель кафедры ИиУС АмГУ
Любовь Александровна Соловцова,
ст. преподаватель кафедры ИиУС АмГУ

ЯЗЫК ПРОГРАММИРОВАНИЯ С++ В ПРИМЕРАХ И ЗАДАЧАХ. Учебно-методическое пособие.

Изд-во АмГУ. Подписано к печати 30.07.02.Формат 60x84/16. Усл. печ. л. 2,32, уч.-изд. л. 2,5. Тираж 100. Заказ 95.