Министерство науки и высшего образования Российской Федерации
Национальный исследовательский университет «МИЭТ»

Н.Ю. Соколова

Практикум по программированию на языке C++ в среде разработки программ MS Visual Studio 2015

Часть 2

Утверждено редакционно-издательским советом университета

Москва 2018

УДК 004.9(07)

Рецензент канд. техн. наук *А.Г. Балашов*

Соколова Н.Ю.

Практикум по программированию на языке C++ в среде разработки программ MS Visual Studio 2015: Часть 2. - М.: МИЭТ, 2018. -120с.: ил.

Рассмотрены задачи программирования на языке C++ в среде Visual Studio 2015 с использованием строкового и комбинированного типов данных, файловых потоков, классов, языка C++/CLI.

Для студентов, обучающихся по направлению «Прикладная информатика». Может быть полезен студентам, обучающимся по другим образовательным программам и для самообразования.

Лабораторная работа № 1

Программирование задач с использованием динамических структур данных

***Цель работы*:** изучение возможности программирования задач с использованием динамических структур; получение практических навыков программирования задач с односвязными списками.

Теоретические сведения

Динамические структуры данных имеют следующие характеристики:

- **непостоянство и непредсказуемость размера** (числа элементов) структуры в процессе ее обработки. Число элементов динамической структуры может изменяться от нуля до некоторого значения, определяемого спецификой соответствующей задачи или доступным объемом машинной памяти;
- **отсутствие физической смежности элементов структуры в памяти**. Логическая последовательность элементов задается в явном виде с помощью одного или нескольких указателей, или связок, хранящихся в самих элементах.

Одним из примеров динамических структур данных является **односвязный список.**

Состав односвязного списка. Он состоит из двух (или более) полей, одно из которых хранит адрес следующего элемента, в качестве информационной части может быть данное любого типа: целого, вещественного, символьного, строкового, структуры и т.п. Схематично взаимосвязь элементов односвязного списка можно представить в следующем виде:

Элемент списка на языке С++ может быть описан так:

```
   struct node{
      int a;           // информационное поле
      struct node* next;/* указатель на следующий эле-
мент */
   };
```

Доступ к элементам списка. Для доступа к элементу списка используется оператор ->:

Переменная-указатель_на_элемент_списка -> имя_поля

Например: node *p; p->a;

Создание списка. При создании списка необходимо выделить память в динамической области под каждый элемент списка и связать элементы между собой. Список можно сформировать следующими способами:

- по принципу **стека**: *первый* созданный элемент будет *последним* в списке,
- по принципу **очереди**: *первый* созданный элемент будет *первым* в списке.

Формирование списка по принципу стека. Алгоритм при формировании списка по принципу стека следующий.

1. Задать указатель на ранее созданный элемент *pred* = NULL (т.е. до формирования списка нет такого элемента).
2. Выделить память под текущий элемент списка ***p*.**
3. Занести значение в информационное поле.
4. Занести значение в поле *next*, равное *pred***.**
5. Сохранить в *pred* значение *p* текущего элемента.
6. Повторять пункты 2 – 5, пока не будет сформировано нужное количество элементов списка.

В *p* будет указатель на первый элемент списка.

Функция формирования односвязного списка имеет вид

```
struct node{
              int a;
              struct node* next;
       };
typedef struct node *Node_ptr;
Node_ptr input(int n)
{
       Node_ptr p=NULL,pred=NULL;
       for(int i=0;i<n;i++)
       {
              p=new struct node;
              p->a=rand()/1000-100;
              p->next=pred;
              pred=p;
       }
       return p;
}
```

Здесь определен собственный тип *Node_ptr* как указатель на структуру *node*. Параметром функции является количество элементов, которое необходимо создать. Функция возвращает указатель на первый элемент списка. В теле функции информационная часть элемента списка заполняется случайным числом.

Формирование списка по принципу очередь. Последовательность действий при формировании списка по принципу очередь следующая.

1. Задать начальное значение указателя на первый элемент списка, равный *NULL* (т.е. до формирования списка нет такого элемента).
2. Если указатель на первый элемент списка равен *NULL*, выделить память под элемент списка *p* и сохранить адрес в ***head*.**
3. Занести значение в информационное поле.
4. Занести значение в поле *next*, равное ***NULL*.**
5. Сохранить в *pred* значение *p* текущего элемента.
6. Повторять пункты 2 – 5, пока не будет сформировано нужное количество элементов списка.

В *head* будет указатель на первый элемент списка.

Функция формирования односвязного списка имеет вид

```
Node_ptr input_1(int n)
{
   Node_ptr p=NULL,head=NULL, pred=NULL;
   for(int i=0;i<n;i++)
   {
          if(head==NULL)
```

```
        {
              head=new struct node;
              head->a=rand()/100-100;
              head->next=NULL;
              pred=head;
        }
        else
        {
              p=new struct node;
              p->a=rand()/100-100;
              p->next=NULL;
              pred->next=p;
              pred=p;
        }
   }
   return head;
}
```

В данной функции, так же как и в предыдущей, единственным параметром является количество элементов списка, который необходимо сформировать. Функция возвращает указатель на первый элемент списка. В отличие от предыдущей функции здесь осуществляется проверка: создан ли первый элемент списка (*head==NULL)* и если нет, то создается элемент, адрес которого сохраняется в переменной *head.*

Функции обработки списка. Функция вывода содержимого списка на экран имеет вид

```
void print_node(Node_ptr p)
{
    Node_ptr x=p;
    while(x!=NULL)     // пока не достигнут конец
                       //списка
    {
// вывод на экран значения информационного поля
       cout<<x->a<<" ";      x=x->next;   // переход
   // к следующему элементу списка
    }
    cout<<endl;//перевод курсора на новую строку
}
```

В функцию вывода значений списка на экран передается указатель на первый элемент списка. Чтобы его значение не было потеряно, вводится дополнительная переменная x, в которую заносится значение *p.* Далее все действия выполняются с использованием указателя *x.*

Функция сортировки информационных полей списка. Текст функции сортировки имеет вид

```
void sort (Node_ptr p)
{
   Node_ptr x, max,
   y=p;   // указатель на первый элемент
          //неотсортированной части списка
   while(y!=NULL)      // пока не конец списка
   {
          max=y;
          x=y;
          while(x!=NULL)
          {      if(x->a > max->a)max=x;
                 x=x->next;
          }
// обмен значений
          int b=max->a;
          max->a=y->a;
          y->a=b;
          y=y->next;    // переход к следующему
                        //элементу списка
   }
}
```

При написании функции сортировки информационных полей элементов списка (далее - сортировка элементов списка) используется алгоритм сортировки одномерных массивов: весь список разбивается на неотсортированную и отсортированную части. В неотсортированной части ищется наибольший элемент и он меняется местами с элементом, стоящим на первом месте неотсортированной части. Далее указатель на первый элемент в неотсортированной части передвигается к следующему элементу. Так же как и в функции вывода содержимого элементов списка, была введена вспомогательная переменная-указатель, которой присвоено значение указателя на первый элемент списка. Результат работы функции показан на рис.1.

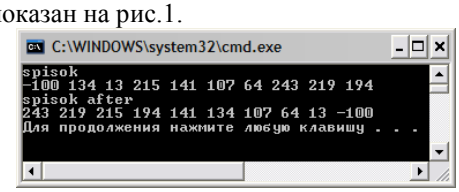


Рис 1. Результат работы функции сортировки односвязного списка

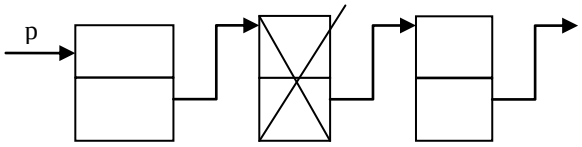
Удаление элемента из списка. Удаление элемента из списка подразумевает:

- изменение связей между элементами, стоящими перед и после удаляемого элемента;
- освобождение памяти из-под удаляемого элемента.

На языке С++ это можно реализовать следующей группой операторов:

```
x=p->next;// сохранить адрес удаляемого элемент
/*записать адрес следующего за удаляемым элементом в
поле next предыдущего элемента */
p->next=p->next->next;
delete x;       //освободить память
```

Схематично это можно представить в следующем виде:



Приведем текст функции удаления элемента после каждого вхождения в список элемента, у которого значение информационного поля равно *E*.

```
void del_node(Node_ptr p,int E)
{
   Node_ptr x=p;
   while(x!=NULL)      // пока не конец списка
   {
    if(x->a==E)//если информационное поле равно E
    {
      if(x->next!=NULL) // если это не последний
                        //элемент
        {
            Node_ptr y=x->next; // сохранить
                        //адрес следующего элемента
/*записать адрес следующего за удаляемым элементом*/
             x->next=x->next->next;
           delete y; // освободить память
        }
   }
   x=x->next;   // перейти к следующему элементу
```

```
            }
}
```

Данная функция не возвращает никакого значения. Параметрами функции являются указатель на первый элемент списка и значение *E*. Результат работы функции показан на рис.2.

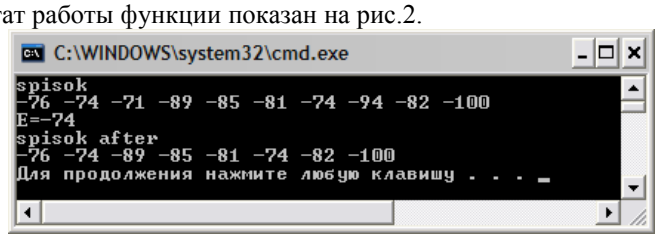


Рис. 2 Результат работы функции удаления элемента из списка

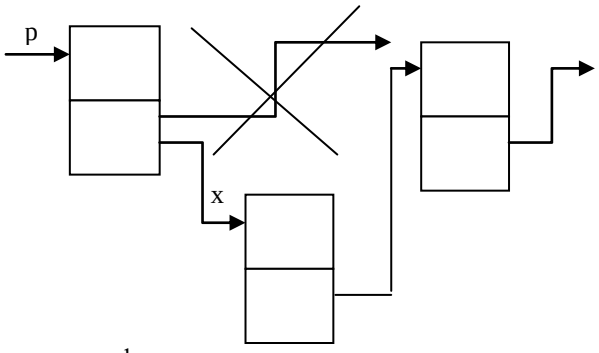
Вставка элемента в список. Вставка элемента в список предполагает:

- выделение памяти для нового элемента;
- изменение имеющихся связей в списке: в поле *next* нового элемента заносится значение указателя на элемент, следующего за текущим элементом, а в поле *next* текущего элемента записывается адрес нового элемента.

На языке C++ эти действия можно представить следующим образом:

```
x=new struct node;     // выделить память
/* записать в поле next нового элемента  значение по-
ля next текущего элемента */
x->next=p->next;
p->next=x; /* записать в поле next текущего элемента
адрес нового элемента */
```

Схематично это можно представить в следующем виде:



Приведем текст функции, которая перед каждым вхождением элемента со значением информационного поля, равным *E*1, вставляет элемент с информационным полем *E*2:

```
Node_ptr insert_node(Node_ptr p, int E1, int E2)
{
   Node_ptr x=p,pred=NULL;
   while(x!=NULL)      // пока не конец списка
   {
          if(x->a==E1) // найден элемент с E1?
          {
                 if(pred==NULL)// если это первый
                              //элемент
                 {
                        Node_ptr n=new node;
                        n->a=E2;
                        n->next=p;
                        p=n;// изменение адреса
                        //первого элемента
                 }
                 else   //это не первый элемент
                 {
                        Node_ptr n=new node;
                        n->a=E2;
                        n->next=x;
                        pred->next=n;
                 }
          }
          pred=x;// сохранить адрес текущего
                 //элемента
          x=x->next;    // перейти к следующему
```

```
                        // элементу
    }
    return p;//возврат указателя на первый элемент
}
```

Функция возвращает указатель на первый элемент списка, поскольку он может измениться, если необходимо будет вставить перед первым элементом исходного списка новый элемент. Для сохранения адреса предыдущего элемента введена переменная *pred*. Результат работы программы с использованием всех функций представлен на рис.3.

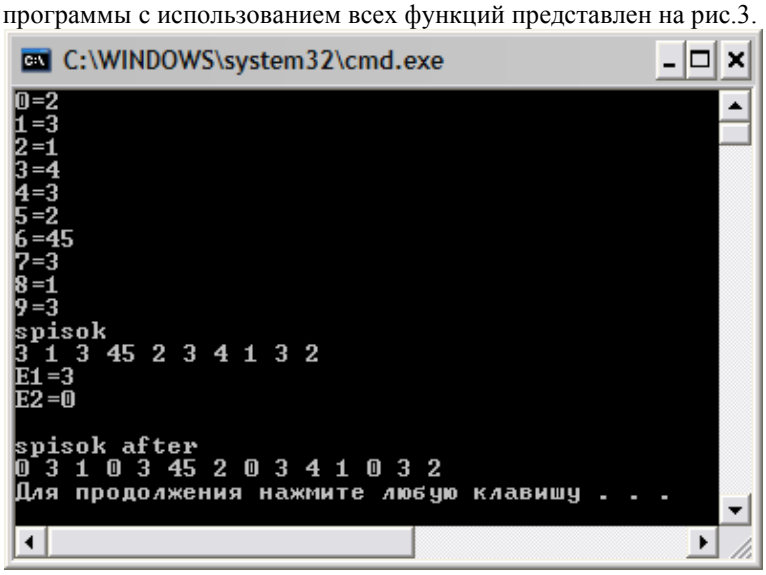


Рис. 3. Результат работы программы в целом

Пояснение к результату работы. Список формировался по принципу стека, значения элементов стека, а также *E1* и *E2* вводились с клавиатуры.

Порядок выполнения работы

1. Разработать и выполнить программу в соответствии с вариантом задания.
2. Результаты выполнения программы занести в отчет по работе.
3. Показать результаты работы преподавателю.

Требования к отчету

Отчет должен содержать:

1) наименование лабораторной работы;
2) краткие теоретические сведения;
3) схему алгоритма;
4) текст программы для своего варианта задания.

Варианты заданий

Написать программу в соответствии с номером варианта. Номер варианта задания соответствует номеру компьютера в компьютерном классе, на котором выполняется лабораторная работа.

Вариант	Задание
1, 16	В составе программы опишите функцию, которая удаляет из списка за каждым вхождением элемента *E*, значение которого введено с клавиатуры, один элемент, если такой есть и он отличен от *E*.
2, 17	В составе программы опишите функцию, которая находит среднеарифметическое значение всех элементов сформированного непустого списка.
3, 18	В составе программы опишите функцию, которая заменяет в списке все вхождения элемента *E1*, значение которого введено с клавиатуры, на элемент *E2*, значение которого также введено с клавиатуры.
4, 19	В составе программы опишите функцию, которая подсчитывает число вхождений элемента *E*, значение которого введено с клавиатуры, в списке *Q*.
5, 20	В составе программы опишите функцию, которая находит наибольший элемент списка и его порядковый номер.
6, 21	В составе программы опишите функцию, которая вставляет в список *K* новый элемент *L1* за каждым вхождением элемента *L*. Значения элементов *L* и *L1* ввести с клавиатуры.

Окончание

7, 22	В составе программы опишите функцию, которая находит среднеарифметическое значение среди элементов списка меньших *P*.
8, 23	В составе программы опишите функцию, которая включает в упорядоченный по убыванию список новое значение, введенное с клавиатуры, таким образом, чтобы не нарушать упорядоченность.
9, 24	В составе программы опишите функцию, которая удаляет все минимальные элементы из списка.
10, 25	В составе программы опишите функцию, которая формирует новый список *L*, включая в него элементы, которые больше среднеарифметического значения элементов списка *L*.
11, 26	В составе программы опишите функцию, которая вставляет в список *Long* за первым вхождением элемента *I*, значение которого введено с клавиатуры, все элементы списка *Short*, если *I* входит в *Long*.
12, 27	В составе программы опишите функцию, которая удаляет из списка каждый *N* элемент.
13, 28	В составе программы опишите функцию, которая в списке *Group* из каждой группы подряд идущих одинаковых элементов оставляет только один элемент.
14, 29	В составе программы опишите функцию, которая удаляет из списка все вхождения элемента E, значение которого введено с клавиатуры.
15, 30	В составе программы опишите функцию, которая меняет местами наибольший и наименьший элемент.

Лабораторная работа № 2

Программирование задач с использованием файловых потоков

***Цель работы*:** изучение различных способов программирования с использованием файловых потоков; получение практических навыков программирования задач с использованием файловых потоков.

Теоретические сведения

Поток – это общий последовательный логический интерфейс с различными устройствами, входящими в состав компьютера (дисплей, клавиатура, принтер, жесткий диск и др.). В общем случае можно сказать, что поток – это логический интерфейс, который связан с файлом, в том числе и с физическим файлом, размещаемым или размещенным в дисковом пространстве компьютера (жестком диске).

Существуют два вида потоков (файлов): текстовый и двоичный (бинарный). Текстовый поток предназначен для ввода-вывода символьных данных, при этом могут происходить некоторые преобразования символов. Например, символ новой строки может быть преобразован при выводе в последовательность из двух символов: возврат каретки и переход на новую строку. В свою очередь, при работе с бинарным потоком никакого преобразования символов не происходит. Введем понятие текущей позиции или указателя потока (файла) – это место в потоке (файле), с которого будут выполняться операции доступа к компонентам потока.

В языке C++ все операции, связанные с файловыми потоками, определены в *fstream*. Поэтому в разделе директив необходимо указать **#include <fstream>.**

Открытие и закрытие файлов. Для открытия файла необходимо его связать с потоком. В языке C++ имеются три вида потоков:

- *ifsteam* – входной поток. Используется только для чтения информации из файла;

- *ofsream* – выходной поток. Используется только для вывода информации в файл;
- *fstream* – двунаправленный поток. Используется для чтения и записи в файл.

Открытие файла выполняется с помощью функции open(). При открытии можно указать способ открытия файла: открытие для дозаписи информации в конец файла, открытие бинарного файла, открытие только для чтения или только для записи и др. Например, если необходимо открыть бинарный файл «text» для чтения и записи, то следует записать:

```
    fstream file;
    file.open("text", ios::binary || ios::in ||
ios::out);
```

Функция *open()* при успешном открытии файла вернет значение TRUE, в противном случае, если файл не будет открыт по каким-то причинам, вернет значение FALSE. Поэтому для проверки открытия файла рекомендуется использовать оператор **if**:

```
    if(!file) cout<<" Не удается открыть файл"; return 1;
```

После работы с файлом его следует закрыть. Особенно это важно, если выполнялись операции записи в файл. Закрытие файла выполняется с помощью функции *close()*: *file.close();*

Чтение и запись данных текстовых файлов. При чтении или записи данных в текстовый файл можно использовать операторы << и >>. Например, для чтения целочисленной переменной *x* из файлового потока *in* необходимо использовать оператор *in>>x,* а для записи в выходной файловый поток *out* значения целочисленной переменной *x*: *out<<x.*

Функция проверки достижения конца файла. Для того чтобы проверить, достигнут ли конец файла, используется функция *eof().* Данная функция возвращает значение TRUE при обнаружении конца файла и FALSE – в противном случае.

Чтение и запись данных бинарных файлов. При работе с бинарными (двоичными) файлами для чтения-записи символов в файл используются функции *get*() и *put*().

Прототипы функций:

```
istream &get(char &ch);
ostream &put(char ch);
```

Функция *get*() считывает из файлового потока символ и помещает его в переменную *ch*. Функция *put*() записывает значение переменной *ch* в файловый поток.

Считывание и запись в файл блоков данных. Для считывания и записи блоков данных используются функции *read*() и *write*().

Прототипы функций:

```
istream &read(char *buf, streamsize num);
ostream &write(const char *buf, streamsize num);
```

Функция *read*() считывает *num* байт данных из связанного с файлом потока в буфер *buf*. Функция *write* () записывает *num* байт данных в связанный с файлом поток из буфера *buf*. Тип *streamsize* является разновидностью целого типа. Он позволяет хранить самое большое количество байтов, которое может быть передано в процессе любой операции ввода-вывода.

Рассмотрим пример, использующий данные функции. Пусть имеется следующая структура данных:

```
struct Tutors
{
   char name[25];
   int years;
   char step[10];
   int stag;
};
```

Информация об основных функциях по работе с файловыми потоками представлена в Приложениях.

В следующем фрагменте программы показано использование функций *read* и *write* при записи и чтении блоков данных, являющихся элементами структуры:

```
Tutors t[10]={
          "Иванов",25,"ktn",3,
          "Петров",45,"no",20,
          "Сидоров",35,"ktn",6,
          "Куликов",58,"dtn",30,
          "Морозов",45,"ktn",22,
          "Малышева",43,"no",10,
          "Мартынова",50,"ktn",28,
          "Яров",60,"ktn",35,
          "Мышкин",39,"no",17,
          "Филиппова",45,"ktn",18
```

```
        };

. . .
ofstream out;
out.open(name, ios::binary);
if(!out){cout<<"файл для чтения "<<name<<" не
открыт"<<endl; return 1;}
out.write((char* ) &t,sizeof t);
out.close();
ifstream in;
in.open(name, ios::binary);
if(!in){cout<<"файл для чтения "<<name<<" не
открыт"<<endl; return 1;}
in.read((char* ) &t,sizeof t);
cout<<setw(26)<<"Name|"<<setw(9)<<"Vozrast|"<<setw(9)
<<"Stepen|"<<setw(7) <<"Stag|"<<endl;
for(int i=0;i<10;i++)
{

cout<<setw(51)<<"_____________________________"<<endl;

cout<<setw(25)<<t[i].name<<"|"<<setw(8)<<t[i].years<<
"|"<<setw(8)<<
   t[i].step<<"|"<<setw(6)<<t[i].stag<<"|";
   cout<<endl;
}
in.close();
```

Отметим, что в `out.write((char* ) &t,sizeof t);`и в `in.read((char* ) &t,sizeof t);` выполнены операции приведения типа `(char* ) &t`, поскольку первый параметр в функциях *read* и *write* представляет собой указатель на символьный массив. Оператор *sizeof* вычисляет размер переменной, следующей за ним. В данном случае размер массива *t*.

Рассмотрим пример, когда имеется бинарный файл, каждый компонент которого является структурой *Tutors*. Тогда чтение данных из бинарного файла с последующим выводом на экран имеет следующий вид:

```
ifstream in;
in.open(name, ios::binary);
if(!in){cout<<"файл для чтения "<<name<<" не
открыт"<<endl; return 1;}
```

```
cout<<setw(26)<<"Name|"<<setw(9)<<"Vozrast|"<<
setw(9)<< "Stepen|"<<setw(7) <<"Stag|"<<endl;
Tutors a;
while(!in.eof())       // пока не конец файла
{
in.read((char* ) &a,sizeof a); /* читать из файла */
cout<<setw(51)<<"_____________________________"<<endl;

cout<<setw(25)<<a.name<<"|"<<setw(8)<<a.years<<"|"
<<setw(8)<<

a.step<<"|"<<setw(6)<<a.stag<<"|";
      cout<<endl;
}
in.close();
```

В отличие от предыдущего фрагмента программы в данной части описана переменная *a* типа *Tutors*, и пока не будет достигнут конец файла, в переменную *a* считываются данные из бинарного файла и выводятся на экран.

Произвольный доступ к компонентам файла. Произвольный доступ можно организовать *только для бинарных файлов*. Произвольный доступ к компонентам файла можно выполнить с помощью функций *seekg*() и *seekp*().

В языке C++ имеется два типа указателей файлов: *get* - указывает на позицию файла при вводе данных; *put* – на позицию при выводе данных.

Прототипы функций:

```
istrean &seekg( off_type  offset, seekdir origin);
istrean &seekp( off_type  offset, origin);
```

Тип *off_type* – это целочисленный тип, позволяющий хранить самое большое значение, которое может принимать параметр *offset*. Тип *seekdir* – это перечисление, принимающее следующие значения:

ios::beg – начало файла;
ios::cur – текущая позиция;
ios::end – конец файла.

Функция *seekg*() перемещает текущий *get*-указатель на *offset* байт относительно позиции, заданной *origin*.

Функция *seekp*() перемещает текущий *put*-указатель на *offset* байт относительно позиции, заданной *origin*.

Пример использования функций . *seekg* и *seekp:*

```
fstream file; // описание файлового потока
file.open(name,ios::binary | ios::out |ios::in);
int i,j;
char b,c;
file.seekg(i);
file.get(b);
file.seekp(i);
file.put(c);
file.seekp(j);
file.put(b);
```

Пример программирования. Информация о преподавателях хранится в текстовом файле. На основе имеющейся информации в текстовом файле создать бинарный файл, компоненты которого представляют собой структуру данных. Вывести на экран сведения о преподавателях в алфавитном порядке, а также список преподавателей от *c* до *d* лет и список преподавателей, чей стаж работы больше *k* лет.

Содержимое исходного текстового файла *data.txt* представлено на рис.1.

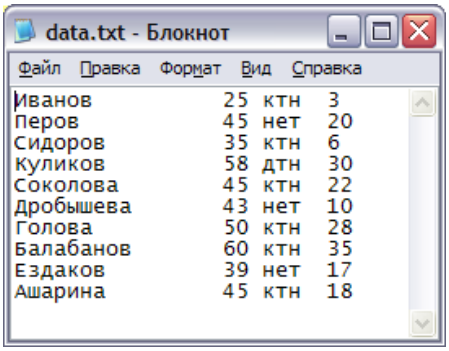


Рис. 1. Исходный текстовый файл *data.txt*

Текст программы:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <clocale>
using namespace std;
// структура «Преподаватели»
```

```
    struct Tutors
    {
       char name[25];
       int years;
       char step[10];
       int stag;
    };
    // функция вывода на экран содержимого файла
    void outfile(fstream& f, int n)
    {
       Tutors a;
       f.seekg(0);/*установить указатель файла на начало*/
       cout << setw(26) << "Имя|" << setw(9) << "Возраст|"
<< setw(9) << "Степень|" << setw(7) << "Стаж|" << endl;
       cout << setw(51) <<
"_____" << endl;
       for (int i = 0; i<n; i++)
       {
              // чтение из файла
              f.read((char *)&a, sizeof a);
              cout << setw(25) << a.name << "|" << setw(8)
<< a.years << "|" << setw(8) << a.step << "|" << setw(6)
<< a.stag << "|";
              cout << endl;
       }
    }
    /* функция чтения данных из текстового файла и запись
данных в бинарный файл */
    int read_f(fstream &f, fstream &out)
    {
       int i = 0; Tutors a;
       while (!f.eof())// пока не конец файла
       {
              f >> a.name >> a.years; f >> a.step; f >>
a.stag;
              out.write((char *)&a, sizeof a);
              i++;
       }
       return i;
    }
    /* функция сортировки компонентов бинарного файла в
алфавитном порядке */
    void sort(fstream &f, int n)
    {
```

```
      Tutors min, a;
      int n_min;
      for (int i = 0; i<n; i++)
      {
            f.seekg(i*(sizeof a));
            f.read((char *)&min, sizeof min);
            n_min = i;
            for (int j = i + 1; j<n; j++)
            {
                  f.read((char *)&a, sizeof a);
                  if (strcmp(a.name, min.name)<0)
                  {
                        min = a;
                        n_min = j;
                  }
            }
            f.seekg(i*(sizeof a));
            f.read((char *)&a, sizeof a);
            f.seekp(i*(sizeof a));
            f.write((char *)&min, sizeof min);
            f.seekp(n_min*(sizeof a));
            f.write((char *)&a, sizeof a);

      }
    }
    /* функция формирования нового файла из данных,
попадающих в заданный интервал */
    int IsYears(fstream &f, fstream &f_new, int n, int c,
int d)
    {
      int k = 0;
      Tutors a;
      f.seekg(0); f_new.seekp(0);
      for (int i = 0; i<n; i++)
      {
            f.read((char *)&a, sizeof a);
            if (a.years >= c && a.years <= d)
            {
                  k++;
                  f_new.write((char *)&a, sizeof a);
            }
      }
      return k;
    }
```

```
    /* функция формирования нового файла из данных, в
которых стаж больше с */
    int IsStag(fstream &f, fstream &f_new, int n, int c)
    {
      int k = 0;
      Tutors a;
      f.seekg(0); f_new.seekp(0);
      for (int i = 0; i<n; i++)
      {
            f.read((char *)&a, sizeof a);
            if (a.stag >= c)
            {
                  k++;
                  f_new.write((char *)&a, sizeof a);
            }
      }
      return k;
    }

    int main()
    {
      setlocale(LC_CTYPE, "Russian");
      fstream in("data.txt", ios::in);
      if (!in)
      {
            cout << "не открыт data.txt" << endl;
            return 1;
      }
      fstream out("data.dat", ios::out | ios::binary);
      if (!out)
      {
            cout << "не открыт data.dat" << endl;
            return 1;
      }
      int size_f = read_f(in, out);
      in.close(); out.close();
      fstream out_f("data.dat", ios::in | ios::out |
ios::binary);
      if (!out_f)
      {
            cout << "не открыт data.dat" << endl;
            return 1;
      }
      sort(out_f, size_f);
```

```
        outfile(out_f, size_f);
        fstream out_f1("data.dat", ios::in | ios::out |
ios::binary);
        if (!out_f1)
        {
                cout << "не открыт data.dat" << endl;
                return 1;
        }
        cout << "Введите через пробел возрастной интервал
>";
        int c, d;
        cin >> c; cin >> d;
        outfile(out_f1, IsYears(out_f, out_f1, size_f, c,
d));
        cout << "Введите стаж>";
        cin >> c;
        outfile(out_f1, IsStag(out_f, out_f1, size_f, c));
        out_f.close(); out_f1.close();
        return 0;
    }
```

Результат выполнения программы показан на рис. 2.

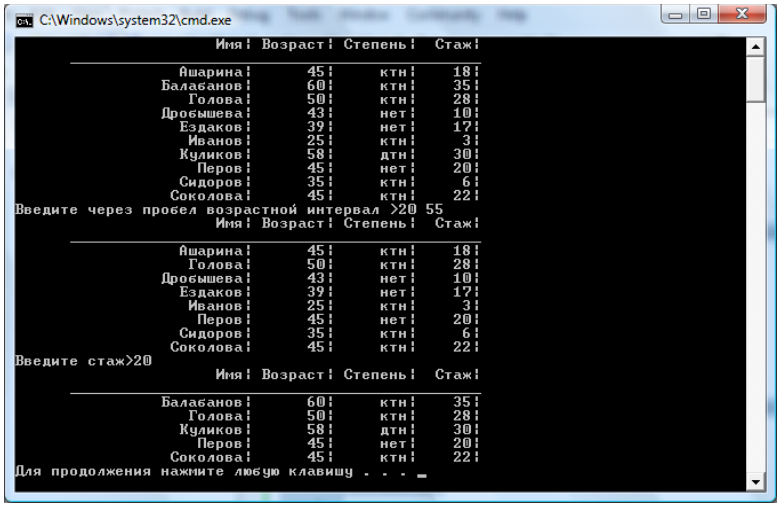


Рис. 2. Результат работы программы по совместному использованию бинарного и текстового файлов

Порядок выполнения работы

1. Разработать и выполнить программу в соответствии с вариантом задания.
2. Результаты выполнения программы занести в отчет по работе.
3. Показать результаты работы преподавателю.

Требования к отчету

Отчет должен содержать:

1) наименование лабораторной работы;
2) краткие теоретические сведения;
3) схемы алгоритмов;
4) текст программы для своего варианта задания.

Варианты заданий

Написать программу в соответствии с номером варианта. Номер варианта задания соответствует номеру компьютера в компьютерном классе, на котором выполняется лабораторная работа.

Вариант	Задание
1	Создайте текстовый файл, содержащий сведения о сотрудниках института: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке возрастания года рождения. Выведите на экран информацию о сотрудниках, чей оклад не превышает *X*.

Продолжение

Вариант	Задание
2	Создайте текстовый файл, содержащий сведения о пациентах глазной клиники: фамилия пациента, пол, возраст, место проживания (город), диагноз. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке фамилий пациентов. Выведите на экран информацию об иногородних пациентах с диагнозом *J*.
3	Создайте текстовый файл, содержащий сведения об ассортименте игрушек в магазине: название игрушки, артикул, цена, страна-производитель, для кого предназначена игрушка (мальчикам, или девочкам, или тем и другим). Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания цены и выведите на экран информацию об игрушках для категории *X* и страны-производителя *Y*.
4	Создайте текстовый файл, содержащий сведения об отправлении поездов дальнего следования с Казанского вокзала: номер поезда, станция назначения, время отправления, время в пути, наличие билетов. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания времени в пути. Выведите на экран информацию о поездах, отправляющихся после *X* часов.

Продолжение

Вариант	Задание
5	Создайте текстовый файл, содержащий сведения о том, какие из пяти предлагаемых дисциплин по выбору желает изучать студент: фамилия студента, факультет, номер группы, пять дисциплин, средний балл успеваемости. Выбираемую дисциплину отметьте символом 1, иначе – 0. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания среднего балла. Выведите на экран дисциплину, которую не желает изучать большинство студентов.
6	Создайте текстовый файл, содержащий сведения о теннисистах чемпионата: фамилия спортсмена, число выигранных матчей, количество двойных подач, количество подач на вылет. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания подач на вылет. Выведите на экран информацию о спортсменах, количество выигранных матчей у которых не менее *X*.
7	Создайте текстовый файл, содержащий сведения о сдаче студентами сессии: факультет, номер группы, фамилия студента, оценки по пяти экзаменам. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке факультетов и выведите на экран информацию о неуспевающих студентах факультета *X*.

Продолжение

Вариант	Задание
8	Создайте текстовый файл, содержащий сведения об ассортименте обуви в магазине фирмы: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д – для дамской обуви, М – для мужской, П – для детской. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке наименований. Выведите на экран информацию об обуви, стоимость которой находится в диапазоне от *A* до *B*.
9	Создайте текстовый файл, содержащий сведения о личной коллекции книголюба: шифр книги, автор, название, год издания, местоположение (номер стеллажа). Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке названий книг и выведите на экран информацию о книгах, изданных после XXXX года.
10	Создайте текстовый файл, содержащий сведения о телефонах абонентов: фамилия абонентов, год установки телефона. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла по возрастанию годов установки телефонов и определите количество установленных телефонов с XXXX года. Номер года вводится с клавиатуры.

Продолжение

Вариант	Задание
11	Создайте текстовый файл, содержащий информацию о рейсах аэропорта (номер рейса, тип самолета, пункт отправления, пункт назначения, время отправления, время прилета). Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке названий пунктов назначений. Выведите на экран все рейсы, время отправления которых позже времени *A*, введенного с клавиатуры.
12	Создайте текстовый файл, содержащий сведения о клиентах фирмы: наименование организации, дата основания, количество договоров на приобретение товаров, общая стоимость договоров. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке увеличения количества договоров. Выведите на экран информацию о фирмах. Общая стоимость договоров находится в интервале от *A* до *C*.
13	Создайте текстовый файл, содержащий сведения о клиентах турагентства: фамилия клиента, страна турпоездки, год поездки, стоимость тура. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания года. Выведите на экран информацию о клиентах, чей тур не превосходил стоимость *Y*.
14	Создайте текстовый файл, содержащий сведения о клиентах фирмы: наименование организации, дата основания, количество договоров на приобретение товаров. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке наименований фирм. Выведите на экран информацию о фирмах, с которыми заключено более *n* договоров.

Продолжение

Вариант	Задание
15	Создайте текстовый файл, содержащий сведения о пациентах глазной клиники: фамилия пациента, пол, возраст, место проживания (город), диагноз. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке мест проживания. Выведите на экран информацию о пациентах, чей возраст попадает в интервал от *A* до *B*.
16	Создайте текстовый файл, содержащий сведения о сотрудниках института: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке возрастания стажа работы. Выведите на экран информацию о сотрудниках, чей стаж работы больше *X* лет.
17	Создайте текстовый файл, содержащий сведения об отправлении поездов дальнего следования с Казанского вокзала: номер поезда, станция назначения, время отправления, время в пути, наличие билетов. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке увеличения времени отправления. Выведите на экран информацию о поездах со станцией назначения *X*.

Продолжение

Вариант	Задание
18	Создайте текстовый файл, содержащий сведения о том, какие из пяти предлагаемых дисциплин по выбору желает изучать студент: фамилия студента, факультет, номер группы, пять дисциплин, средний балл успеваемости. Выбираемую дисциплину отметьте символом 1, иначе – 0. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке фамилий. Выведите на экран информацию о студентах, желающих изучать дисциплину *X*, чей средний балл успеваемости не ниже *Y*.
19	Создайте текстовый файл, содержащий сведения о нападающих команды «Спартак»: фамилии игроков, число заброшенных ими шайб, число сделанных голевых передач, заработанное штрафное время. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания забитых голов. Выведите на экран информацию о нападающих, имеющих результативные передачи не менее *X*.
20	Создайте текстовый файл, содержащий сведения об ассортименте обуви в магазине фирмы: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д – для дамской обуви, М – для мужской, П – для детской. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке убывания стоимости. Выведите на экран информацию о всей дамской обуви.

Продолжение

Вариант	Задание
21	Создайте текстовый файл, содержащий сведения о клиентах турагентства: фамилия клиента, страна турпоездки, год поездки, стоимость тура. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в порядке возрастания стоимости тура. Выведите на экран информацию о клиентах, посетивших страну *X*.
22	Создайте текстовый файл, содержащий сведения о личной коллекции книголюба: шифр книги, автор, название, год издания, местоположение (номер стеллажа). Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке авторов книг и выведите на экран информацию о книгах автора *X*.
23	Создайте текстовый файл, содержащий сведения о сдаче студентами сессии: факультет, номер группы, фамилия студента, оценки по пяти экзаменам. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке фамилий неуспевающих студентов и выведите на экран средний балл, полученный каждым студентом группы *X*.
24	Создайте текстовый файл, содержащий сведения об ассортименте игрушек в магазине: название игрушки, артикул, цена, страна производитель, для кого предназначена игрушка (мальчикам или девочкам или тем и другим). Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке стран-производителей игрушек и выведите на экран информацию об игрушках по стоимости, не превышающей *X* рублей для категории *Y*.

Окончание

Вариант	Задание
25	Создайте текстовый файл, содержащий сведения о телефонах абонентов: фамилия абонентов, год установки телефона. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке фамилий абонентов и по вводимой фамилии абонента выведите его номер телефона.
26	Создайте текстовый файл, содержащий информацию о рейсах аэропорта: номер рейса, тип самолета, пункт отправления, пункт назначения, время отправления, время прилета. Напишите программу, которая формирует двоичный файл, каждый элемент которого является структурой, составленной на основе данных текстового файла. Отсортируйте данные бинарного файла в алфавитном порядке названий пунктов отправления. Выведите на экран все рейсы, отправляющиеся в пункт **A**. Значение *A* введите с клавиатуры.

Лабораторная работа № 3

Программирование задач с использованием классов

Цель работы: изучение возможности программирования с использованием классов; получение навыков программирования с использованием абстрактного типа данных.

Теоретические сведения

Класс – это абстрактный тип данных языка программирования. При определении класса описываются данные класса и действия, которые выполняются над этими данными. Данные принято называть членами класса, а действия – функциями-членами или методами.

Объявление класса. Общий вид объявления класса:

class *имя_класса* **{**
закрытые_члены_и_методы;
public:
открытые_члены_и_функции
} *список_объектов***;**

Объявление класса похоже на объявление структуры за исключением встраивания в класс методов и того, что данные являются открытыми по умолчанию, а в классе по умолчанию все данные закрытые (*private*).

Пример объявления класса «Преподаватель»:

```
class tutor
{
   char *name;
   int years;
   char *step;
   int stag;
public:
   void init(char *,int,char *,int);
```

```
    void print();
    char * get_name();
    int get_years();
};
```

Членами класса *tutor* являются: *name* – фамилия преподавателя, *years* – возраст, *step* – степень и *stag* – стаж работы. Эти члены являются закрытыми (по умолчанию). В качестве методов используются: *init* – инициализация данных; *print* – вывод на экран всех значений класса; *get_name* – возврат фамилии; *get_years* – возврат возраста. Для того чтобы методами можно было воспользоваться в основной программе, доступ к ним открывается с помощью ключевого слова – спецификатора *public*. В классе указывается, как правило, только прототип метода, сам метод определяется вне класса. В программе может быть объявлено несколько классов, в которых имеются методы с одинаковыми именами и с одинаковым списком параметров. Для того чтобы компилятор смог разобраться, к какому методу функция относится, применяется оператор разрешения области видимости **::**

Приведем определение одного из методов класса *tutor:*

```
void tutor::init(char  *n, int g, char *s,
int st)
{
          name=new char[strlen(n)+1];
          strcpy(name,n);
          years=g;
          step=new char[strlen(s)+1];
          strcpy(step,s);
          stag=st;
}
```

Доступ к данным и методам класса. Общий вид:

имя_объекта**.**имя_данного
имя_объекта**.**имя_метода(список фактических параметров)

Например:

```
tutor a;
char* name="Иванов";
int god=43, st=15;
char *step="ктн";
a.init(name,god,step,st);
```

Конструкторы и деструкторы. ***Конструктор*** – это функция, которая вызывается при создании объекта, т.е. выполняется операция инициализации объекта. Конструктор – это специальный метод класса, имя которого совпадает с именем класса. Конструкторы не возвращают никаких значений. Конструктор объекта вызывается при создании объекта, т.е. при объявлении объекта.

Пример конструктора для класса «односвязный список»:

```
struct Node
{
   int    n;
   struct Node *next;
};
class list
{
   Node *First;/* указатель на первый элемент списка*/
   int count;   // количество элементов списка
public:
         list(int k);// конструктор
...
}
list::list(int k)
{
   First=new Node;
   First->n=k;
   First->next=NULL;
   count=1;
};
```

В основной программе при определении объекта с использованием конструктора необходимо записать: *list a(number);* где *number* – это переменная, имеющая некоторое значение. Например, *int number=rand()/1000-10;*

Деструктор – это функция, которая вызывается при разрушении объекта. Например, освобождается память, которая была выделена при инициализации в конструкторе. Имя деструктора совпадает с именем класса, но перед ним ставится символ тильда ~ . Так же как и конструкторы, деструкторы не возвращают значения.

Приведем объявление класса с использованием деструктора:

```
struct Node
{
   int    n;
   struct Node *next;
```

```
};
class list
{
   Node *First;
   int count;
public:
          list(int k);
~list();
};
list::list(int k)
{
   First=new Node;
   First->n=k;
   First->next=NULL;
   count=1;
};
list::~list()
{
   Node *start=First,*follow=First;
   while(start!=NULL)
   {

          follow=start;
          start=start->next;
          delete follow;

   }
};
```

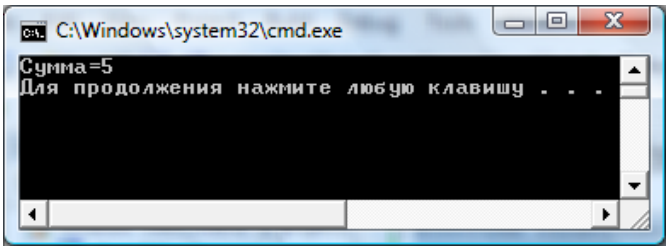
Дружественные функции. В языке C++ имеется возможность разрешить доступ к закрытым членам класса через функции, не являющиеся членами класса. Такие функции называют дружественными. Для разрешения доступа включить прототип функции в описание открытой части класса (public) и перед прототипом функции указать ключевое слово *friend*:

```
class cl{
…
public:
friend void frnd(int a);
};
```

Рассмотрим пример, в котором дружественная функция имеет доступ к закрытым членам класса:

```
class numbers
{
   int a;
   int b;
public:
   numbers(int a1,int b1){a=a1,b=b1;};
   friend int sum(numbers a);
};
int sum(numbers s)
{
   return s.a+s.b;
};
В основной программе:
#include <iostream>
#include <clocale>
#include "frien.h"
using namespace std;
int main()
{
   setlocale(LC_CTYPE,"Russian");
   numbers a=numbers(2,3);/* инициализация данных
через конструктор */
   cout<<"Сумма="<<sum(a)<<endl;
}
```

Результат выполнения программы показан на рис.1.



Рие.1. Разультат выполнения программы с использованием дружественной функции

Отметим, что при определении дружественной функции не применяется оператор области видимости **::**, а при обращении к функции *sum* не используется оператор «точка».

Перегрузка конструкторов. Язык C++ позволяет использовать несколько конструкторов в зависимости от параметров. Например, значения объекта могут в конструкторе создаваться по умолчанию либо с

использованием данных, передаваемых через параметры (параметризованный конструктор), или объект создается как копия другого объекта. В последнем случае такой конструктор называется копирующим. Приведем фрагмент программы с конструктором по умолчанию и параметризованным конструктором.

```
class list
{
   Node *First;
   int count;
public:
   list();       // конструктор по умолчанию
   list(int k);// параметризованный конструктор
   ~list();      // деструктор
};
list::list()
{
   First=new Node;
   First->n=0;
   First->next=NULL;
   count=1;
};
list::list(int k)
{
   First=new Node;
   First->n=k;
   First->next=NULL;
   count=1;
};
```

Фрагмент основной программы:

```
int number=rand()/1000-10;
list a(number);// параметризованный конструктор
list c;// конструктор по умолчанию
```

Копирующий конструктор или конструктор копии. Копирующий конструктор, или конструктор копии, – это специальный тип перегруженного конструктора.

Приведем реализацию копирующего конструктора для класса «список»:

```
class list
{
```

```
   Node *First;
   int count;
public:
   list(int k);
   list(const list &list1);
   ~list();
};
list::list(const list &list1)
{
   First=new Node;
   Node *l=list1.First;
   First->n=l->n;;
   First->next=NULL;
   l=l->next;
   Node *sled,*pred=First;
   while(l!=NULL)
   {
          sled=new Node;
          pred->next=sled;
          sled->n=l->n;
          sled->next=NULL;
          l=l->next;
          pred=sled;
   }
   count=list1.count;
}
```

Ключевое слово this. Слово this – это указатель на объект, который вызвал данный метод. При каждом вызове метода ему автоматически передается указатель *this* – неявный параметр, принимаемый всеми методами. Покажем использование *this* при объявлении класса *my*:

```
class my
{
   int p;
public:
   my();// конструктор по умолчанию
   my(int a);// параметризованный конструктор
   void set(int i){this->p=i;};/* метод изменения
значения p */
int get(){return this->p;};// вернуть значение p
};
my::my()
{
```

```
   this->p=0;
};
```

Перегрузка операторов. В языке C++ имеется возможность перегрузки операторов. Перегружая оператор, можно определить его значение для конкретного класса. Например, для класса «односвязный список» можно определить оператор «+» для присоединения в конец имеющегося списка другого списка, а оператор «– –» – для удаления последнего элемента из списка. Перегруженный оператор для данного класса работает как совершенно новый оператор. Чтобы перегрузить оператор, необходимо определить значение новой операции для класса, к которому она будет применяться.

Общий формат функции *operator*:

```
тип имя_класса::operator #
(список_формальных_параметров);
{
   реализация оператора
}
```

Операторная функция может быть как членом класса, так и не членом, а дружественной функцией. Приведем в качестве примера перегрузку оператора «+» для класса «список». Данный оператор позволит осуществить операцию конкатенации: к одному из списков добавить копию другого списка. Именно копию, а не изменение указателя последующего элемента одного списка на значение первого элемента второго списка, что приведет к прерыванию выполнения программы.

Пример программы с перегрузкой оператора «+»:

```
using namespace std;
struct Node
{
   int    n;
   struct Node *next;
};
class list
{
   Node *First;
   int count;
public:
   list(int k);
   list(const list &list1);
   list& operator +(const list &k);
```

```
void      add(int);
void      print();
int               number_list(){return count;};
void      sub();
~list();
};
```

Представим реализацию только оператора «+»

```
list& list::operator +(const list &k)
{
   Node *start=First,*another=k.First;
   while(start->next!=NULL)
          start=start->next;
   while(another!=NULL)
   {
   Node *new_Node=new Node;
   new_Node->n=another->n;
   new_Node->next=NULL;
   start->next=new_Node;
   start=start->next;
   another=another->next;
   count++;
   }
   return *this;

};
```

В основной программе:

```
int main()
{
   setlocale(LC_CTYPE,"Russian");
   srand((unsigned)time(NULL));
   int number=rand()/1000-10;
   list a(number);
   cout<<"Список a"<<endl;
   for(int i=0; i<10; i++)
   {
          number=rand()/1000-10;
          a.add(number);
   }
   a.print();
   list b(number);
   cout<<"Список b"<<endl;
```

```
        for(int i=0; i<10; i++)
        {
               number=rand()/1000-10;
               b.add(number);
        }
        b.print();
        a+b;
        cout<<"Список a+b"<<endl;
        a.print();
        cout<<"Количество элементов списка a=" <<
a.number_list()<<endl;
    }
```

Результат выполнения программы показан на рис. 2.

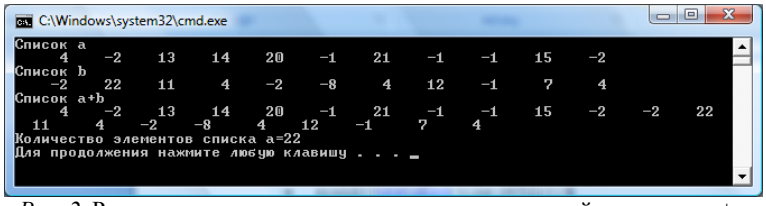


Рис. 2. Результат выполнения программы с перегрузкой оператора «+»

Перегрузка унарных операторов. Рассмотрим перегрузку унарных операторов на примере перегрузки оператора «++», который добавляет в конец списка новый элемент, значение информационного поля которого – случайное число:

```
class list
{
   Node *First;
   int count;
public:
   list(int k);
   list(const list &list1);
   list& operator++();
~list();
};
list& list::operator++()
{
   Node *start=First;
   while(start->next!=NULL)
          start=start->next;
   Node *new_Node=new Node;
```

```
   int k=rand()/1000-10;
   new_Node->n=k;
   new_Node->next=NULL;
   start->next=new_Node;
   count++;
   return *this;
};
```

Фрагмент основной программы:

```
   ++c;
   cout<<"Список с после ++"<<endl;
   c.print();
```

На перегрузку операторов налагается ряд ограничений: нельзя изменить приоритет оператора и количество операндов, операторные функции не могут иметь параметра по умолчанию. Есть операторы, которые нельзя перегружать: :: . .* ?

С использованием дружественных функций нельзя перегружать:

= () [] –>

Пример программирования. В качестве примера программирования приведем программу работы с односвязными списками, в которой реализованы два конструктора: параметризованный и конструктор копии, а также деструктор, методы добавления элемента в список, вывода на экран элементов списка, определения количества элементов списка, удаления элемента из списка, перегруженные операторы конкатенации, присваивания и добавления элемента в конец списка.

Содержимое файла list.h с определением класса «список» имеет вид

```
using namespace std;
struct Node
{
   int    n;
   struct Node *next;
};
class list
{
   Node *First;
   int count;
public:
         list(int k);
         list(const list &list1);
         list& operator +(const list &k);
```

```
        list &operator =(const list &k);
        list& operator++();
void    add(int);
void    print();
int             number_list(){return count;};
void    sub();
~list();
};
list::list(int k)
{
   First=new Node;
   First->n=k;
   First->next=NULL;
   count=1;
};
list::list(const list &list1)
{
   First=new Node;
   Node *l=list1.First;
   First->n=l->n;;
   First->next=NULL;
   l=l->next;
   Node *sled,*pred=First;
   while(l!=NULL)
   {
          sled=new Node;
          pred->next=sled;
          sled->n=l->n;
          sled->next=NULL;
          l=l->next;
          pred=sled;
   }
   count=list1.count;
}
void list::add(int k)
{
   Node *start=First;
   while(start->next!=NULL)
          start=start->next;
   Node *new_Node=new Node;
   new_Node->n=k;
   new_Node->next=NULL;
   start->next=new_Node;
   count++;
```

```
};
list& list::operator++()
{
   Node *start=First;
   while(start->next!=NULL)
          start=start->next;
   Node *new_Node=new Node;
   int m=rand()/1000-10;
   new_Node->n=m;
   new_Node->next=NULL;
   start->next=new_Node;
   count++;
   return *this;
};
list& list::operator +( const list &k)
{
   Node *start=First,*another=k.First;
   while(start->next!=NULL)
          start=start->next;
   while(another!=NULL)
   {
   Node *new_Node=new Node;
   new_Node->n=another->n;
   new_Node->next=NULL;
   start->next=new_Node;
   start=start->next;
   another=another->next;
   count++;
   }
   return *this;
};
void list::print()
{
   Node *start=First;
   while(start!=NULL)
   {
          cout<<setw(6)<<start->n;
          start=start->next;

   }
   cout<<endl;
};
void list::sub()
```

```
{
   Node *start=First,*the_end;
   for(int i=0;i<count-2;i++)
          start=start->next;
   the_end=start->next;
   start->next=NULL;
   delete the_end;
   count--;
};
list::~list()
{
   Node *start=First,*follow=First;
   while(start!=NULL)
   {

          follow=start;
          start=start->next;
          delete follow;

   }
};
list &list::operator =( const list &k)
{
   this->First=new Node;
   Node *l=k.First;
   this->First->n=l->n;;
   this->First->next=NULL;
   l=l->next;
   Node *sled,*pred=First;
   while(l!=NULL)
   {
          sled=new Node;
          pred->next=sled;
          sled->n=l->n;
          sled->next=NULL;
          l=l->next;
          pred=sled;
   }
   this->count=k.count;
   return *this;
};
```

Текст основной программы (файл list.cpp):

```
#include <iostream>
```

```
#include <iomanip>
#include <time.h>
#include "list.h"
using namespace std;
int main()
{
	setlocale(LC_CTYPE, "Russian");
	srand((unsigned)time(NULL));
	int number = rand() / 1000 - 10;
	list a(number);
	cout << "Список a" << endl;
	for (int i = 0; i<10; i++)
	{
		number = rand() / 1000 - 10;
		a.add(number);
	}
	a.print();
	list b(number);
	cout << "Список b" << endl;
	for (int i = 0; i<10; i++)
	{
		++b;
	}
	b.print();
	a + b;
	cout << "Список a+b" << endl;
	a.print();
	cout << "Количество элементов списка a=" <<
a.number_list() << endl;
	list c(number);
	for (int i = 0; i<10; i++)
	{
		number = rand() / 1000 - 10;
		c.add(number);
	}
	cout << "Список c" << endl;
	c.print();
	c = a;
	cout << "Список c после присваивания" << endl;
	c.print();
	++c;
	cout << "Список c после ++" << endl;
	c.print();
}
```

Результат работы программы приведен на рис. 3.

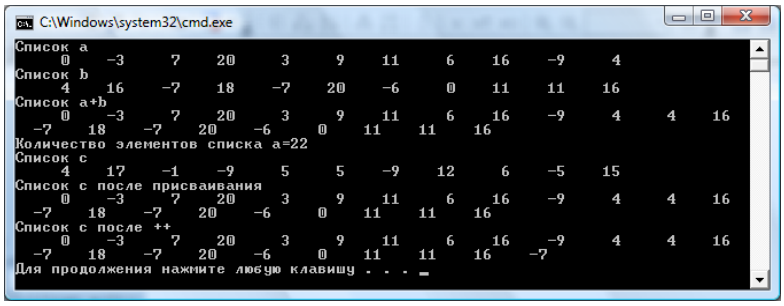


Рис.3. Результат работы программы с односвязными списками

Порядок выполнения работы

1. Разработать и выполнить программу в соответствии с вариантом задания.
2. Результаты выполнения программы занести в отчет по работе.
3. Показать результаты работы преподавателю.

Требования к отчету

Отчет должен содержать:

1) наименование лабораторной работы;
2) краткие теоретические сведения;
3) схемы алгоритмов;
4) текст программы для своего варианта задания.

Варианты заданий

Написать программу из блока заданий № 1 или № 2 (с использованием перегрузки операторов) в соответствии с номером варианта. Номер варианта задания соответствует номеру компьютера в компьютерном классе, на котором выполняется лабораторная работа.

Блок заданий № 1

Вариант	Задание
1	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, получение значения элемента массива по его порядковому номеру, нахождение минимального элемента.
2	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение суммы положительных элементов матрицы, получение значения элемента массива по его индексам.
3	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки, нахождение самого короткого слова в строке.
4	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции члены: вывод на экран элементов списка, поиск максимального элемента.
5	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, получение значения элемента массива по его порядковому номеру, нахождение произведения элементов, значения которых по модулю меньше 5.
6	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк и конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение наименьшего элемента матрицы среди положительных элементов, получение значения элемента массива по его индексам.

Продолжение

Вариант	Задание
7	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки, конкатенации двух строк.
8	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка, нахождение среднеарифметического значения отрицательных элементов.
9	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, сортировки элементов массива по возрастанию.
10	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение количества элементов матрицы, попадающих в заданный интервал [k, m].
11	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки, перевод символов строки в верхний регистр.
12	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции члены: вывод на экран элементов списка, сортировки элементов списка по убыванию.
13	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, поиск максимального элемента среди элементов, попадающих в интервал [k, m$)$.

Продолжение

Вариант	Задание
14	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение количества элементов матрицы, не отличающихся от первого элемента более чем на 6.
15	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки, перевод символов строки в нижний регистр.
16	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка, вставки нового элемента *E2* после элемента *E1*.
17	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, поиск количества элементов, меньших значения *P*, введенного с клавиатуры.
18	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение среднеарифметического среди элементов главной диагонали.
19	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки, количество символов, равных символу *S*, введенному с клавиатуры.
20	Определите класс «список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции члены: вывод на экран элементов списка, удаления элемента после элемента *E*.

Окончание

Вариант	Задание
21	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, нахождение среднеарифметического среди отрицательных элементов.
22	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение суммы положительных элементов главной диагонали.
23	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки, нахождение количества слов в строке.
24	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка, нахождение порядкового номера минимального элемента.
25	Определите класс «стек». В класс включите два конструктора: создание стека (пустого) ; конструктор-копию. Определите функции-члены: вывод на экран элементов стека, помещение в стек элемента и удаление элемента из стека.

Блок заданий № 2

Вариант	Задание
1	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, получение значения элемента массива по его порядковому номеру. Пергрузите оператор: – – нахождение минимального элемента.

Продолжение

Вариант	Задание
2	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, получение значения элемента массива по его индексам. Перегрузите оператор: + получение новой матрицы, каждый элемент которой равен сумме элементов двух других матриц
3	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки. Перегрузите оператор: – – нахождение самого короткого слова в строке.
4	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов и конструктор-копия. Определите функции-члены: вывод на экран элементов списка. Перегрузите оператор: [ ] поиск максимального элемента.
5	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива, получение значения элемента массива по его порядковому номеру. Перегрузите оператор: – получение нового массива, каждый элемент которого равен разности элементов двух массивов.
6	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы, нахождение наибольшего элемента матрицы среди положительных элементов. Перегрузите оператор: [ ] получения значения элемента массива по его индексам.

Продолжение

Вариант	Задание
7	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки. Перегрузите оператор: + конкатенация двух строк.
8	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка. Перегрузите оператор: – – нахождение среднеарифметического значения отрицательных элементов.
9	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива. Перегрузите оператор: ++ сортировка элементов массива по возрастанию.
10	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы. Перегрузите оператор: [ ] нахождение количества нулевых элементов матрицы.
11	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки. Перегрузите оператор: ++ перевод символов строки в верхний регистр.
12	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка. Перегрузите оператор: – – сортировка элементов списка по убыванию.

Продолжение

Вариант	Задание
13	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива. Перегрузите оператор: ++ поиск максимального элемента.
14	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы. Перегрузите оператор: + нахождение количества элементов матрицы, не отличающихся от первого элемента более чем на 6.
15	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки. Перегрузите оператор: –– перевод символов строки в нижний регистр .
16	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка. Перегрузите оператор: + вставка нового элемента *E2* после элемента *E1*.
17	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива. Перегрузите оператор: – поиск количества элементов, меньших значения *P*, введенного с клавиатуры.

Продолжение

Вариант	Задание
18	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы. Перегрузите оператор: ++ нахождение среднеарифметического среди элементов главной диагонали.
19	Определите класс «строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки,. Перегрузите оператор: – нахождение количества символов, равных символу *S*, введенному с клавиатуры.
20	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка. Перегрузите оператор: – удаление элемента после элемента *E*.
21	Определите класс «одномерный массив». В класс включите два конструктора: создание одномерного массива по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов массива. Перегрузите оператор: – – нахождение среднеарифметического среди отрицательных элементов.
22	Определите класс «матрица». В класс включите два конструктора: создание матрицы по количеству столбцов и строк; конструктор-копию. Определите функции-члены: вывод на экран элементов матрицы. Перегрузите оператор: ++ нахождение суммы положительных элементов главной диагонали.

Окончание

Вариант	Задание
23	Определите класс 2строка символов». В класс включите два конструктора: создание строки символов; конструктор-копию. Определите функции-члены: вывод на экран строки. Перегрузите оператор: – – нахождение количества слов в строке.
24	Определите класс «односвязный список». В класс включите два конструктора: создание списка по количеству элементов; конструктор-копию. Определите функции-члены: вывод на экран элементов списка. Перегрузите оператор: – – нахождение порядкового номера минимального элемента.
25	Определите класс «стек». В класс включите два конструктора: создание стека (пустого); конструктор-копию. Определите функции-члены: вывод на экран элементов стека. Перегрузите операторы: + помещение в стек элемента – – удаление элемента из стека.

Лабораторная работа № 4

Программирование на языке C++/CLI

Цель работы: изучение особенностей программирования на C++/CLI; получение практических навыков программирования в среде CLR Visual Studio 2005.

Теоретические сведения

В данной лабораторной работе исполбзуются следующие сокращения:

CLI – Common Language Infrastructure – инфраструктура общего языка;

CLR – стандартизованная среда выполнения программ, написанных на VB, C#, C++.;

CLI – это, по сути, спецификация среды виртуальной машины, которая позволяет приложениям, написанным на различных высокоуровневых языках программирования, выполняться в различных системах без изменения и перекомпиляции исходного кода;

CLI – спецификация стандарта. CLR – реализация CLI от Microsoft.

Типы данных. Соответствие стандартных типов данных языкаC++ типам данных языка C++/CLI приведено в табл.1.

Таблица 1

Соответствие стандартных типов данных C++/CLI

C++ (стандартный)	C++/CLI
bool	System::Boolean
char	System::Sbyte
int	System::Int32
long	System::Int32
float	System::Single
double	System::Double
wchat_t	System::Char

Чтобы не использовать при описании данных *System::*, можно воспользоваться *using namespace System*;

Ввод данных с клавиатуры. Для ввода данных с клавиатуры используются функции *Console::Read()* и *Console::ReadLine()*. Функция *Console::Read()* возвращает в качестве значения символ, считанный с клавиатуры, а возвращаемое значение функции *Console::ReadLine()* является строкой символов.

Примеры использования функций:

```
char a=Console::Read();
String^ line= Console::ReadLine();          // чтение строки символов
```

Вывод данных на экран. Вывод на экран можно реализовать с помощью функций *Console::WriteLine* или *Console::Write*. Первая функция после вывода данных переведет курсор на следующую строчку, а последняя нет.

Примеры использования функций:

```
Console::WriteLine(L"Текст");/* вывод с переводом курсора на новую строку */
Console::Write(L"Текст");      /* вывод без перевода курсора на новую строку*/
Console::WriteLine(L"Текст {0} еще текст {1,5:F2} ",a,b);
Console::WriteLine(L"Текст {0} еще текст {1:F2} ",a,b);
```

Символ *L* перед строкой, заключенной в кавычки, означает, что строка состоит из «широких» символов, где каждый символ занимает 2 байт.

Рассмотрим подробнее вывод данных с помощью двух последних строк.

Console::WriteLine(L"Текст {0} еще текст {1,5:F2} ",a,b); в данном операторе вместе с текстом будут выведены значения переменных *a* и *b*. Значение переменной *a* будет подставлено вместо *{0}*, а при выводе значения переменной *b* будет использовано форматирование. Запись *{1,5:F2}* означает, что значение выводится с десятичной точкой, причем общее число позиций равно 5, а две позиции отводятся на цифры после десятичной точки. 0 и 1 после открывающей фигурной скобки определяют порядковый номер переменных, перечисленных после закрывающей кавычки и запятой. Нумерация начинается с нуля.

Отслеживаемые дескрипторы. Динамическое выделение памяти в CLR работает иначе, чем в консольном приложении. CLR поддерживает свою собственную «кучу» памяти, которая полностью независима

от «кучи» языка C++. CLR автоматически очищает память, которая была выделена и необходимость в которой отпала, что позволяет не использовать *delete* для освобождения памяти. Время от времени CLR упорядочивает память для избежания фрагментации. Процесс автоматического возвращения памяти в «кучу» и упорядочивания памяти называется *сборкой «мусора»*.

Для выделения памяти в C++/CLI используется операция *gcnew*. CLR отслеживает каждую переменную, ссылающуюся на участок памяти в «куче», и когда обнаруживается, что переменная уже не существует, то происходит автоматическое освобождение памяти. Процесс упорядочивания памяти подразумевает изменения адресов для ранее выделенных участков памяти, поэтому указатели стандартного языка C++ здесь применяться не могут. Способ доступа к объектам в «куче» обеспечивается *отслеживаемыми дескрипторами* и отслеживаемыми *ссылками* (аналог ссылок языка C++).

Отлеживаемый дескриптор хранит адрес, который автоматически обновляется сборщиком «мусора». Нельзя применять арифметику адресов к дескрипторам, как это делается в языке C++ при работе с указателями. Кроме этого, не разрешается приведение дескрипторов. Все объекты, созданные в «куче» CLR, должны снабжаться дескрипторами. Все объекты ссылочных типов классов сохраняются в «куче», и потому созданные программистом переменные, которые ссылаются на такие объекты, должны быть дескрипторами. Например, класс *String* – это ссылочный тип класса, поэтому переменные, которые ссылаются на объекты *String*, должны быть отслеживаемыми дескрипторами.

Общий вид объявления отслеживаемых дескрипторов:

имя_класса^ имя_переменной

Например,

String^ stroka;

Для установки дескриптора в ноль служит *nullptr*:

stroka= nullptr;

Можно явно инициализировать дескриптор набором символов:

String^ stroka=L"Это строка символов";

Приведение типов. Очень часто при программировании с использованием языка C++/CLI применяется приведение типов.

Общий вид:

safe_cast *тип*(*имя_переменной*)

Приведем пример использования приведения типов:

```
double a=5.6;
double b=7.8;
int s=safe_cast<int>(a)+safe_cast<int>(b);// s=5+7
```

Отметим, что с клавиатуры можно ввести только символ или строку символов. В случае необходимости чтения с клавиатуры целого или вещественного значения следует прочитать строку и воспользоваться функциями преобразования строки символов в целое значение:

`Convert::ToInt32(строковая_переменная)` или строки символов в вещественное значение

`Convert::ToDouble(строковая_переменная)`

Например:

```
String^ line;
double a=Convert::ToDouble(line);
int b= Convert::ToInt32(line);
```

Примеры программирования. Приведем примеры программирования задач, рассмотренных в лабораторных работах № 1–8 части 1[1].

Программирование линейных алгоритмов

Пример 1. Написать программу для вычисления математического выражения $y = \sqrt{tg^2 ln\, ab^2 + a^2 sin^2 e^{2x+5}}$

Текст программы:

```
#include "stdafx.h"
#include <math.h>
using namespace System;
int main(array<System::String ^> ^args)
{
  double a,b,x,y;
   String^ line;
   Console::Write(L"a>");
   line=Console::ReadLine();
   a=Convert::ToDouble(line);
```

[1] *Соколова Н.Ю. Практикум по программированию на языке С++ в среде разработки программ MS Visual Studio 2015: Часть 1. – М.: МИЭТ, 2017. – 160с.*

```
        Console::Write(L"b>");
        line=Console::ReadLine();
        b=Convert::ToDouble(line);
        Console::Write(L"x>");
        line=Console::ReadLine();
        x=Convert::ToDouble(line);
        y=sqrt((pow(tan(log(a*b*b)),2))+a*a*pow
(sin(exp(2*x+5)),2));
        Console::WriteLine(L"a={0} b= {1} x={2}", a,b,x);
        Console::WriteLine(L"y={0,4:F2}  ",y);
          return 0;
    }
```

Пояснения к программе. В данной программе вводятся с клавиатуры значения *a,b,x*, которые используются при вычислении значения *y*. Вычисленное значение *y* и значения *a,b,x* выводятся на экран.

Результат выполнения программы показан на рис.1.

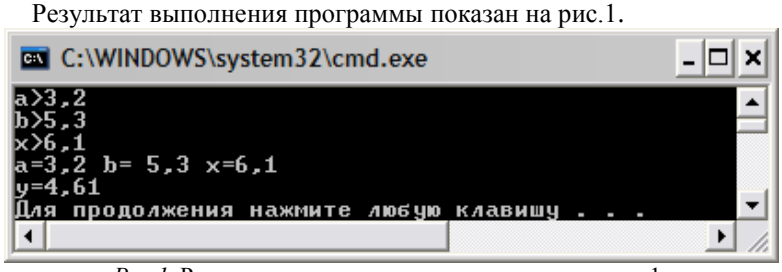


Рис.1. Результат выполнения программы к примеру 1.

Обратите внимание, вещественные значения вводятся и выводятся с использованием запятой между целой и дробной частями числа.

Программирование ветвящихся алгоритмов

Пример 2. Вычислить значение

$$Y=\begin{cases}\sqrt{tg^2 lnab^2+a^2 sin^2 e^{2x+5}} \text{ при } x<ab\\ tg^2\ln|ax^2+b^4|+arctg(a+b) \text{ при } x\geq ab\end{cases}$$

Текст программы:

```
#include "stdafx.h"
#include <math.h>
using namespace System;
int main(array<System::String ^> ^args)
{
```

```
        double      a,b,x,y;
       String^ line;
       Console::Write(L"a>");
       line=Console::ReadLine();
       a=Convert::ToDouble(line);
       Console::Write(L"b>");
       line=Console::ReadLine();
       b=Convert::ToDouble(line);
       Console::Write(L"x>");
       line=Console::ReadLine();
       x=Convert::ToDouble(line);
       if(x<a*b) y = sqrt(pow(tan(log(a*b*b)),
2)+a*a*pow(sin(exp(2*x+5)), 2));
        else y = pow(tan(log(fabs(a*x*x + pow(b,4))
)),2)+atan(a+b);
       Console::WriteLine(L"y={0,4:F3}  ",y);
        return 0;
    }
```

Результат выполнения программы показан на рис.2.

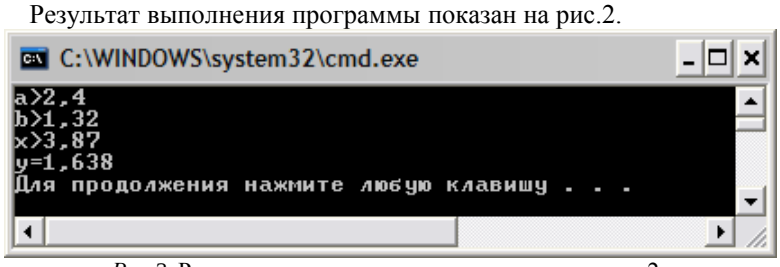


Рис.2. Результат выполнения программы к примеру 2.

Пример 3. По введенной отметке-цифре вывести ее название. Например, 5 – отлично.

Текст программы:

```
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{
   Console::Write(L"Оценка>");
   char c=Console::Read();
   switch(c)
     {
        case '1':case '2':
          Console::WriteLine(L"Неуд>");break;
        case '3':
```

```
Console::WriteLine(L"Удовлетворительно>");break;
       case '4':
         Console::WriteLine(L"Хорошо>");break;
       case '5':
         Console::WriteLine(L"Отлично>");break;
     default:Console::WriteLine(L"Ошибка>");
      } }
```

Результат выполнения программы показан на рис.3.

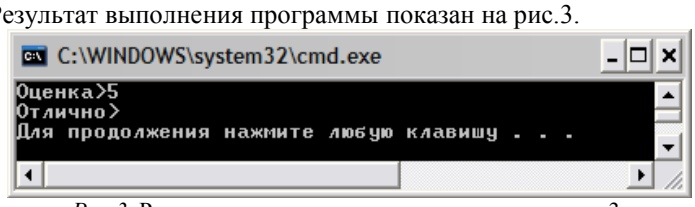


Рис.3. Результат выполнения программы к примеру 3

Программирование циклических алгоритмов

В языке C++/CLI имеется еще одни оператор цикла *for each*
Общий вид:

for each(тип имя_переменной **in** имя_переменной_массива)
{
…тело цикла
}

Работает оператор следующим образом: для каждого значения, входящего в массив, выполняются операторы тела цикла.

Пример 4. Подсчитать количество гласных и согласных букв латинского алфавита в строке, введенной с клавиатуры.

Текст программы:

```
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{
   Console::Write(L"Строка>");String ^
c=Console::ReadLine();
   int gl=0,sogl=0, prob=0;
   for each(wchar_t sym in c)
   {
         switch(sym) {
         case 'a':case 'e':
```

```
            case 'i':case 'u':
            case 'o':gl++;break;
            case ' ':prob++;break;
                  default:sogl++;       }
      }
      Console::WriteLine(L"Гласных={0}",gl);
      Console::WriteLine(L"Согласных= {0}",sogl);
   }
```

Пояснения к программе. С клавиатуры вводится строка символов. Количество гласных накапливается в переменной *gl*, количество согласных - в *sogl*, количество пробелов – в переменной *prob*. Начальные значения этих переменных равны нулю. Далее в цикле *for each* для каждого символа исходной строки осуществляется анализ текущего символа. Если этот символ гласная, то увеличивается значение переменной *gl* на единицу, если пробел, то увеличивается на единицу значение количества пробелов, в противном случае увеличивается на единицу значение переменной *sogl*.

Результат выполнения программы приведен на рис.4.

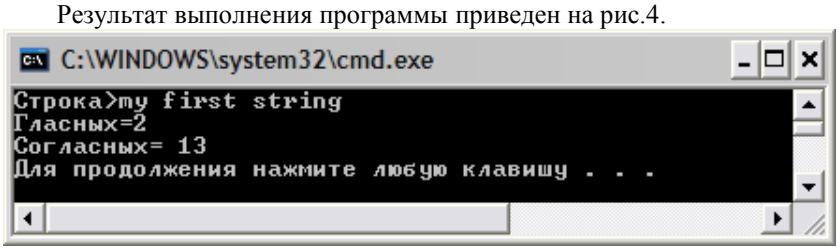


Рис.4. Результат выполнения программы к примеру 4

Программирование задач с использованием одномерных массивов

Общий вид описания одномерного массива:

array<int>^ имя_массива=*gcnew array<int>*(кол-во_эл_массива);

Пример 5. Сформировать одномерный массив с использованием датчика случайных чисел. Отсортировать элементы массива по возрастанию их значений.

Текст программы:

```
#include "stdafx.h"
using namespace System;
```

```
int main(array<System::String ^> ^args)
{
	array<int>^ mas = gcnew array<int>(15);
	Random^ rnd = gcnew Random;
	Console::WriteLine(L"Массив до сортировки");
	for (int i = 0; i<mas->Length; i++) mas[i] = rnd-
>Next() / 10000000 - 100;
	for each(int a in mas) Console::Write(L"{0}  ", a);
	Console::WriteLine();
	Array::Sort(mas);
	Console::WriteLine(L"Массив после сортировки");
	for each(int a in mas) Console::Write(L"{0}  ", a);
	Console::WriteLine();
	return 0;
}
```

Пояснения к программе. Размер массива `mas` определяется с помощью метода *Length: mas->Length*. Для присваивания случайных значений элементам массива применяется метод *Next: rnd ->Next()*. Сортировка элементов массива осуществляется с использованием метода *Sort Array::Sort(mas);* (т.е. в данной программе не реализуется свой метод сортировки).

Результат выполнения программы показан на рис.5.

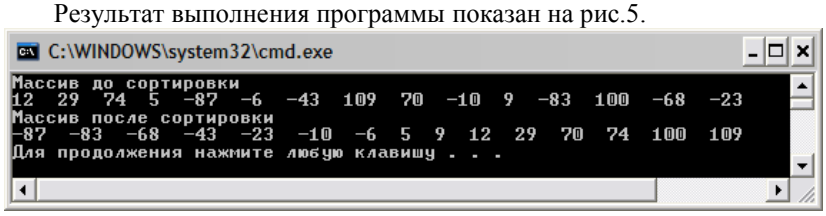


Рис.5. Результат выполнения программы к примеру 5

Программирование задач с использованием многомерных массивов

Общий вид описания многомерного массива:

*array<int, 2>^* имя_массива=*gcnew array<int, 2>* (кол-во_строк, кол-во_столбцов);

Пример 6. Сформировать одномерный массив из элементов массива *F* размерностью 4 на 5, которые попадают в диапазон [*d;l*].

Текст программы:

```
#include "stdafx.h"
using namespace System;
const int n=4,m=5;
int main(array<System::String ^> ^args)
{
   array<int>^ mas =gcnew array<int>(n*m);
   array<int,2>^ matr=gcnew array<int,2>(n,m);
   Random^ rnd=gcnew Random;
   Console::WriteLine("Матрица:");
   for(int i=0;i<n;i++)
   {
          for(int j=0;j<m;j++)
          {
            matr[i,j]=rnd->Next()/10000000-100;
            Console::Write(L"{0,4}  ",matr[i,j]);
          }
          Console::WriteLine();
   }
   String^ line;
   int d,l;
   Console::Write(L"d>");
   line=Console::ReadLine();
   d=Convert::ToDouble(line);
   Console::Write(L"l>");
   line=Console::ReadLine();
   l=Convert::ToDouble(line);
   int k=0;
   for(int i=0;i<n;i++)
   {
          for(int j=0;j<m;j++)
            if(matr[i,j]>=d && matr[i,j]<=l)
            {
                 mas[k]=matr[i,j];
                 k++;
            }
   }
   Console::WriteLine("Массив:");
   for(int i=0;i<k;i++) Console::Write(L"{0}  ",
mas[i]);
   Console::WriteLine();
    return 0;
}
```

Результат выполнения программы показан на рис.6.

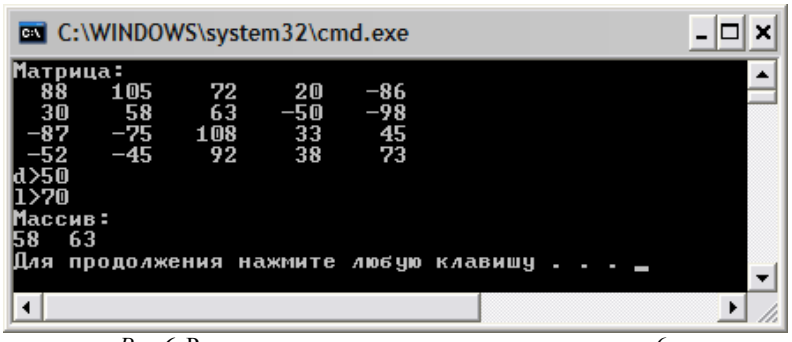


Рис.6. Результат выполнения программы к примеру 6

Программирование задач с использованием функций

Пример 7. Написать программу для задания примера 6 с помощью функций: создания матрицы, формирования одномерного массива, вывода значений одномерного массива на экран.

Текст программы:

```
#include "stdafx.h"
using namespace System;
const int n = 4, m = 5;
// функция формирования матрицы
void input(array<int, 2>^ a, Random^ rnd)
{
	for (int i = 0; i<n; i++)
	{
		for (int j = 0; j<m; j++)
		{
			a[i, j] = rnd->Next() / 10000000 -
100;
			Console::Write(L"{0,4}  ", a[i, j]);
		}
		Console::WriteLine();
	}
}
// функция формирования одномерного массива
int form(array<int, 2>^ a, array<int>^ b,
	int d, int l)
{
	int k = 0;
	for (int i = 0; i<n; i++)
```

```
	{
		for (int j = 0; j<m; j++)
			if (a[i, j] >= d && a[i, j] <= l)
			{
				b[k] = a[i, j];
				k++;
			}
	}
	return k;
}
// функция вывода одномерного массива на экран
void print(array<int>^ b, int k)
{
	for (int i = 0; i<k; i++)
		Console::Write(L"{0}  ", b[i]);
	Console::WriteLine();
}
// основная функция
int main(array<System::String ^> ^args)
{
	Random^ rnd = gcnew Random;
	array<int>^ mas = gcnew array<int>(n*m);
	array<int, 2>^ matr = gcnew array<int, 2>(n, m);
	Console::WriteLine("Матрица:");
	input(matr, rnd);
	String^ line;
	int d, l;
	Console::Write(L"d>");    line = Con-
sole::ReadLine();
	d = Convert::ToDouble(line);
	Console::Write(L"l>");
	line = Console::ReadLine();
	l = Convert::ToDouble(line);
	int k = form(matr, mas, d, l);
	Console::WriteLine("Массив:");
	print(mas, k);
	return 0;
}
```

Результат работы программы показан на рис.7.

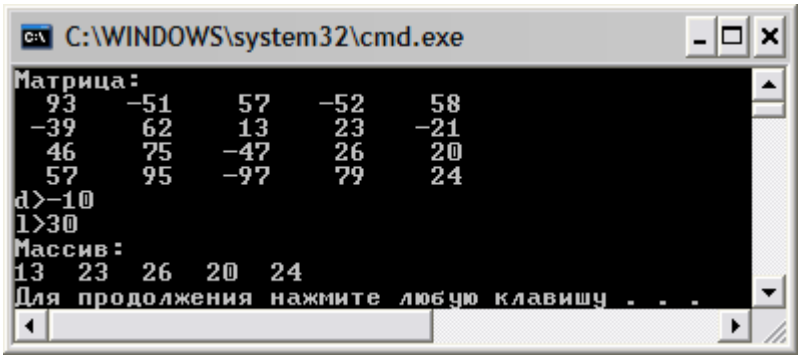


Рис.7. Результат работы программы к примеру 7

Программирование задач с использованием строк

Объявление и инициализация строковых переменных:

```
    String^ line=L"Это строка";  /* строка состоит из ши-
роких символов*/
    String^ line="Это строка";/* строка состоит из 8-
битовых символов */
```

Определение длины строки

```
    String^ line=L"Это строка";
    line->Length;   // длина строки
```

Конкатенация двух строк

```
    String^ line1=L"Это строка";
    String^ line2=L"это еще одна строка";
    String^ line3=line1+L" и  "+ line2;/*объединение
строк */
```

В табл. 2 приведены некоторые функции работы со строками.

Таблица 2

Основные функции работы со строками

Функция	Физический смысл функции
Trim()	Удаление пробелов или символов, определенных пользователем в начале и конце строки
TrimStart()	Удаление в начале строки
TrimEnd()	Удаление в конце строки
PadLeft() PadRight()	Добавление символов слева Добавление символов справа

Окончание

Функция	Физический смысл функции
ToUpper()	Преобразование символов строки в верхний регистр
ToLower()	Преобразование символов строки в нижний регистр
Insert(n,line)	Вставка строки *line* с позиции *n*
Replace(l1,l2)	Заменить подстроку *l*1 на *l*2
StartWith(line)	Начинается ли строка *line* подстрокой?
EndWith(line)	Заканчивается ли строка line подстрокой?
IndexOf(sym)	Номер позиции первого вхождения символа *sym* или подстроки, начиная с начала строки
LastIndexOf(sym)	Номер позиции символа или подстроки *sym*, начиная с конца строки

Пример 8. Дана строка символов. Вывести на экран четыре самых коротких слова строки.

Текст программы:

```
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{
   Console::Write(L"Введите строку:");
   String^ line=Console::ReadLine();
   line=line+" ";//добавление пробела в конец строки
   array<String^>^ word=gcnew array<String^>(30);
   int pos=0,i=0;
   int pos_prob=line->IndexOf(' ',pos);/* поиск
позиции пробела */
   while(pos_prob>0)// пока пробелы в строке есть
   {
/* выделение слова с первой позиции до позиции
пробела */
word[i++]=line->Substring(pos,pos_prob-pos);
/* позиция с которой начнется поиск = позиции пробела
+1 */
pos=pos_prob+1;
// поиск нового слова до пробела
         pos_prob=line->IndexOf(' ',pos);
   }
// сортировка массива слов по длине слова
```

```
        array <int>^ l_w=gcnew array<int>(30);
        for(int j=0;j<i;j++) l_w[j]=word[j]->Length;
        Array::Sort(l_w,word);
        Console::WriteLine("Четыре самых коротких слова");
        for(int j=30-i;j<30-
i+4;j++)Console::WriteLine(word[j]);
        return 0;
    }
```

Результат выполнения программы показан на рис.8:

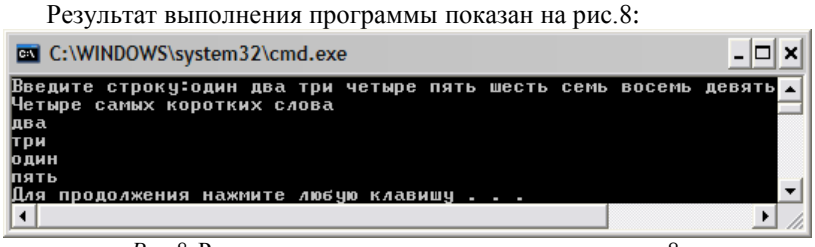


Рис.8. Результат выполнения программы к примеру 8

Создание проекта CLR

Для создания проекта необходимо:

1) в меню *Файл* выбрать *Создать-> Проект*. В появившемся перечне типов проекта (рис.9) выбрать *CLR-> Консольное приложение CLR Visual C++* и ввести имя проекта;

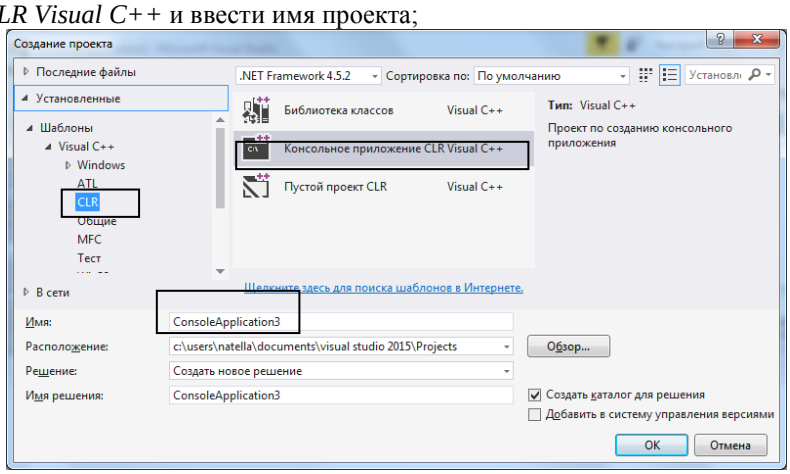


Рис.9. Создание проекта CLR

2) заменить оператор *Console::WriteLine(L"Hello World");* текстом своей программы. Построить проект и запустить программу на выполнение аналогично *Консольному приложению Win32*.

После создания проекта Visual Studio открывает шаблон главной функции (рис.10).

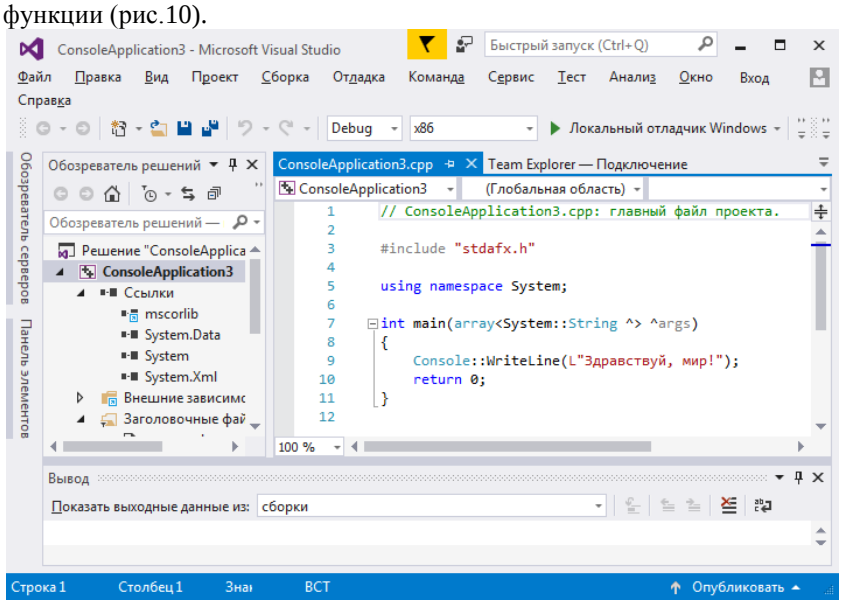


Рис.10. Рабочее поле с шаблоном основной функции

Порядок выполнения работы

1. Разработать и выполнить программу в соответствии с вариантом задания.
2. Результаты выполнения программы занести в отчет по работе.
3. Показать результаты работы преподавателю.

Требования к отчету

Отчет должен содержать:

1) наименование лабораторной работы;
2) краткие теоретические сведения;

3) схемы алгоритмов;
4) текст программы для своего варианта задания.

Варианты заданий

Напишите программу с использованием функций с параметрами на языке C++/CLI в соответствии с номером варианта. Номер варианта задания соответствует номеру компьютера в компьютерном классе, на котором выполняется лабораторная работа.

Вариант	Задание
1	Задайте значения целочисленным элементам массивов *A* (8), *B*(5), *C*(6) и вычислите $S = \sum_{i=0}^{7} a_i + \sum_{j=0}^{4} b_j + \sum_{k=0}^{5} c_k$
2	Задайте значения целочисленным элементам массивов *M*(7), *L*(6), *C*(8) и вычислите $Y = \prod_{i=0}^{5}(l_i - 2) + \prod_{j=0}^{7} c_j + 0{,}5\prod_{i=0}^{5}(m_n - 4)$
3	Задайте значения вещественным элементам массивов *A*(6), *B*(3), *C*(10) и вычислите *Y* = *max*(*max*(A), *max*(B), *max* (C)).
4	Задайте значения целочисленным элементам матриц *A* и *B* размерностью 3 на 6 и сформируйте массивы *C* и D, состоящие из максимальных элементов столбцов матриц *A* и *B* соответственно.
5	Задайте значения вещественным элементам массивов *A* и *B* размерностью 4 на 4 и сформируйте массивы *X* и *Y* из элементов, стоящих на главных диагоналях матриц *A* и *B* соответственно.
6	Задайте значения целочисленным элементам матриц *P* и *Q* размерностью 4 на 8 и сформируйте массивы *C* и *D*, состоящие из максимальных элементов строк матриц *P* и *Q* соответственно.

Продолжение

Вариант	Задание
7	Задайте значения вещественным элементам матриц *A* и *Q* размерностью 2 на 4 и сформируйте массивы *B* и *R*, состоящие из минимальных элементов столбцов матриц *A* и *Q* соответственно.
8	Задайте значения целочисленным элементам матриц *P* и Q размерностью 4 на 7 и сформируйте массивы *R* и *T* из сумм отрицательных элементов строк матриц *P* и *Q* соответственно.
9	Задайте значения вещественным элементам матриц *C* и *D* размерностью 5 на 5 и сформируйте массивы *X* и *Y* из произведений положительных элементов столбцов матриц *C* и *D* соответственно.
10	Задайте значения целочисленным элементам матриц *W* и *Z* размерностью 4 на 6 и сформируйте массивы *T* и *S* соответственно из элементов матриц *W* и *Z*, больших заданного числа *P*.
11	Задайте значения вещественным элементам матриц *B* и *D* размерностью 6 на 8 и сформируйте массивы *Y* и *Z*, состоящие соответственно из элементов матриц *B* и *D*, меньших заданного числа *R*.
12	Задайте значения вещественным элементам матриц *P* и *Q* размерностью 3 на 6 и сформируйте массивы *R* и *T*, состоящие из минимальных элементов столбцов матриц *P* и *Q* соответственно.
13	Задайте значения целочисленным элементам матриц *C* и *D* размерностью 6 на 6 и сформируйте одномерные массивы *S* и *T* соответственно из элементов той строки матрицы, у которой элемент на главной диагонали минимален.
14	Задайте значения вещественным элементам массивов *A*(6), *B*(6), *C*(6) и вычислить $X = min(A) + min(B) - min(C)$.
15	Задайте значения целочисленным элементам матриц *A* и *B* размерностью 4 на 7 и сформируйте одномерные массивы *Y* и *Z* соответственно, каждый элемент которых является суммой отрицательных элементов строк.

Продолжение

Вариант	Задание
16	Задайте значения целочисленным элементам матриц *A* и *B* размерностью 7 на 5 и сформируйте одномерные массивы *X* и *Y* соответственно, каждый элемент которых является произведением элементов строк, попадающих в интервал [*m*, *n*].
17	Задайте значения целочисленным элементам матриц *A* и *B* размерностью 3 на 4 и сформируйте одномерные массивы *X* и *Y*, каждый элемент которых является элементом матрицы кратным *P*.
18	Задайте значения вещественным элементам матриц *B* и *D* размерностью 5 на 3 и сформируйте массивы *Y* и *Z*, состоящие соответственно из элементов матриц *B* и *D*, попадающих в заданный интервал [*d*, *k*].
19	Задайте значения вещественным элементам матриц *P* и *Q* размерностью 4 на 7 и сформируйте массивы *R* и *T*, состоящие из среднеарифметических значений столбцов матриц *P* и *Q* соответственно.
20	Задайте значения целочисленным элементам матриц *M* и *N* размерностью 5 на 7 и сформируйте массивы *C* и *D*, состоящие из количества отрицательных элементов строк матриц *M* и *N* соответственно.
21	Задайте значения вещественным элементам матриц *A* и *Q* размерностью 3 на 5 и сформируйте массивы *B* и *R*, состоящие из количества положительных элементов столбцов матриц *A* и *Q* соответственно.
22	Задайте значения целочисленным элементам матриц *M* и *N* размерностью 6 на 6 и сформируйте массивы *C* и *D*, состоящие из количества элементов строк больших значения элемента, стоящего на главной диагонали, матриц *M* и *N* соответственно.
23	Задайте значения вещественным элементам матриц *A* и *Q* размерностью 5 на 5 и сформируйте массивы *B* и *R*, состоящие из элементов больших среднеарифметического значения элементов главной диагонали.

Окончание

Вариант	Задание
24	Задайте значения целочисленным элементам матриц *P* и *Q* размерностью 6 на 6 и сформируйте массивы *R* и *T* из элементов меньших наибольшего элемента среди элементов главной диагонали матриц P и Q соответственно.
25	Задайте значения вещественным элементам матриц *C* и *D* размерностью 5 на 5 и сформируйте массивы *X* и *Y* из сумм положительных элементов строк матриц *C* и *D* соответственно.

Литература

1. ***Хортон Айвор***. Visual C++ 2005. Базовый курс: пер. с англ. – М.: ООО Изд. дом «Вильямс», 2007.

2. ***Пахомов Б.*** C/C++ и MS Visual C++ 2005 для начинающих.– СПб.: БВХ-Петербург, 2007.

3. ***Шилд Г.*** C++ базовый курс. – М., СПб.: ООО Издат. дом «Вильямс», 2007.

Приложение 1

Библиотечные функции работы со строками

В языке C++ используются следующие библиотечные функции работы со строками:

char* strcat(char * string1, char * string2) - добавляет строку *string2* в конец строки *string1*, записывая в конец строки результата нуль-символ, и возвращает указатель на сцепленную строку (*string1*);

char* strchr (char* string, int sim) - возвращает указатель на первое местонахождение символа, имеющего код *sim*, в строке *string*. Символ *sim* может быть нулевым символом (‘\0’), тогда поиск ведется для нулевого символа. Функция возвращает NULL, если символ не найден;

int strcmp (char* string1, char* string2) - сравнивает строки *string1* и *string2* лексикографически и возвращает значение:

меньше 0, если *string1*< *string2*,
равное 0, если *string1*= *string2*,
больше 0, если *string1*> *string2*;

char* strcpy (char* string1, char* string2) - копирует строку *string2*, включая нуль-символ, в строку *string1* и возвращает значение аргумента *string1*;

int strcspn (char* string1, char* string2) - возвращает индекс первого символа в строке *string1*, который принадлежит набору символов *string2*. Завершающий нуль-символ не учитывается при поиске. Если *string1* начинается с символа из *string2*, то возвращается значение 0;

int strlen (char* string) - возвращает длину в байтах строки *string*. Нуль-символ не учитывается;

char* strncat (char* string1, char* string2, int n) - добавляет первые *n* символов из строки *string2* в строку *string1*, завершая результирующую строку нуль-символом. Если n больше длины строки *string2*, то длина строки *string2* используется вместо *n*;

int strncmp (char* string1, char* string2, int n) - сравнивает первые *n* символов в строках *string1* и *string2* лексикографически и возвращает результат:

значение < 0, если *string1*< *string2*,
= 0, если *string1*= *string2*,
> 0, если *string1*> *string2*;

char* strncpy (char* string1, char* string2, int n) - копирует n символов строки *string2* в строку *string1*. Если значение меньше, чем длина строки *string2*, то нуль-символ не добавляется в новую строку. Если значение *n* больше, чем длина строки *string2*, то нуль-символ добавляется в конец строки *string1*;

char* strpbrk (char* string1, char* string2) - находит первое вхождение в строке *string1* любого символа из набора символов, содержащихся в строке *string2*. Завершающий нуль-символ не включается в поиск. Возвращаемое значение - указатель на первое местоположение любого символа из *string2* в *string1* или значение NULL, если нет общих символов;

int strspn (char* string1, char* string2) - возвращает индекс первого символа строки *string1*, который не принадлежит набору символов, содержащихся в строке *string2*. Нуль-символ не рассматривается. Если строка *string1* начинается с символа не из набора *string2* , функция возвращает значение 0;

char* strstr (char* string1, char* string2) - возвращает указатель на первое вхождение подстроки, которая содержится в символьном массиве *string2* в строке *string1*. Возвращает NULL, если вхождение не найдено;

char* strtok (char* string1, char* string2) - символы из *string1* группируются в слова *string2* – набор символов-разделителей для строки *string1*. При первом вызове *strtok* производит возврат адреса первого символа *string1*. Чтобы найти начало следующего слова в *string1*, необходимо вызвать *strtok* с NULL-значением аргумента *string1*. Набор разделителей может различаться от вызова к вызову. Возвращаемое значение-указатель на слово в строке. Все слова завершаются нуль-символом.

Перечислим функции проверки символов заголовочного файла **ctype.h**.

int isalnum (int c) - проверяет символ с на латинскую букву или цифру. Возвращаемое значение = 0, если это буква или цифра;

int isalpha (int c) - проверяет символ с на латинскую букву;

int isdigit (int c) - проверяет символ с на десятичную цифру.

Приложение 2

Библиотечные функции работы с файловыми потоками

В языке С++ используются библиотечные функции работы с файловыми потоками.

istream &get(char *buf, streansize num) -считывает символы в массив *buf* до тех пор, пока не будет считано *num*-1 символов либо не встретится символ новой строки или символ конца файла. После выполнения функции в массив *buf* будет добавлен символ конца строки (\0);

istream &get(char *buf, streansize num, char delim) - считывает символы в массив *buf* до тех пор, пока не будет считано *num* – 1 символов или не встретится символ, заданный параметром *delim*, либо символ конца файла. После выполнения функции в массив *buf* будет добавлен символ конца строки (\0). Символ, заданный *delim*, в массив *buf* не записывается, он остается во входном потоке до следующей операции ввода;

int get() - возвращает из потока следующий символ;

sreamsize gcount() - возвращает количество символов, считанных при выполнении последней операции ввода;

istream &getline(char *buf, streansize num) - считывает символы в массив *buf* до тех пор, пока не будет считано *num* – 1 символов или не встретится символ новой строки либо символ конца файла. После выполнения функции в массив *buf* будет добавлен символ конца строки (\0). Если встретится символ новой строки, то он в *buf* не помещается, а из входного потока извлекается;

istream &getline(char *buf, streansize num, char delim) - считывает символы в массив *buf* до тех пор, пока не будет считано *num* – 1 символов или не встретится символ, заданный параметром *delim*, либо символ конца файла. После выполнения функции в массив *buf* будет добавлен символ конца строки (\0). Символ, заданный *delim*, в массив *buf* не записывается, но извлекается из входного потока;

int peek() - считывает следующий символ из входного потока, но не удаляет его из него;

istream &putback (char c) - возвращает последний считанный символ из потока в него;

ostream &flush() - немедленно перезаписывает содержимое буферов на жесткий диск, не дожидаясь их (буферов) заполнения. Как пра-

вило, информация накапливается во внутреннем буфере. Пока буфер не будет полностью заполнен, информация на диск не переносится. Эта операция позволяет немедленно переписать данные на диск.

Содержание

Учебное издание

Соколова *Натэлла Юрьевна*

Практикум по программированию на языке C++ в среде разработки программ MS Visual Studio 2015. Часть 2

Редактор *А.В. Тихонова*. Технический редактор *Л.Г. Лосякова*. Корректор *Л.Г. Лосякова*. Верстка автора.

Подписано в печать с оригинал-макета 26.06.2013. Формат 60 × 84 1/16. Печать офсетная. Бумага офсетная. Гарнитура Times New Roman. Усл. печ. л. 2,55. Уч.-изд. л. 2,2. Тираж 150 экз. Заказ .

Отпечатано в типографии ИПК МИЭТ.
124498, г. Москва, г. Зеленоград, площадь Шокина, дом1, МИЭТ.