Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование

Вариант 10

Выполнил:

Студент группы РИС-20-1Б

Ремянников Александр Владимирович

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

Получить опыт работы вобъектно-ориентированном программировании.

## 18.1

**Постановка задачи**

(Вариант 10)

1.  Реализовать определение нового класса. Для демонстрации работы с объектами написать главную функцию. Продемонстрировать разные способы создания объектов и массивов объектов.

2.  Структура-пара – структура с двумя полями, которые обычно имеют имена first и second. Требуется реализовать тип данных с помощью такой структуры. Во всех заданиях должны присутствовать :

a.  метод инициализации Init (метод должен контролировать значения аргументов на корректность);

b.  ввод с клавиатуры Read;

c.  вывод на экран Show.

3.  Реализовать внешнюю функцию make\_тип(), где тип – тип реализуемой структуры. Функция должна получать значения для полей структуры как

параметры функции и возвращать структуру как результат. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

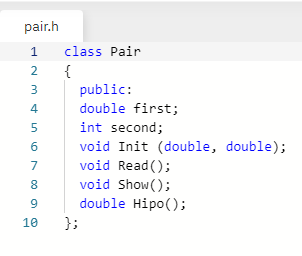
Задание варианта:

Поле first – дробное положительное число, катет a прямоугольного треугольника, поле second – дробное положительное число, катет b прямоугольного треугольника. Реализовать метод hipotenuse() –вычисление гипотенузы.

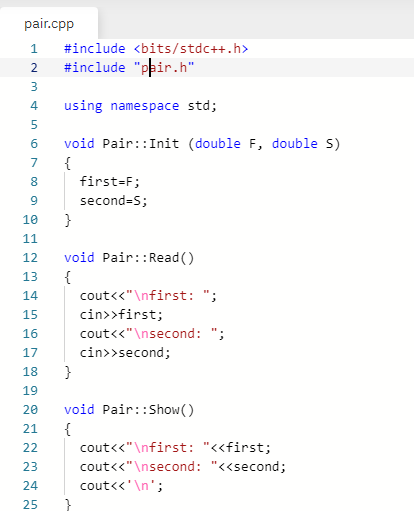
**Анализ задачи**

**1.** Для решения задачи необходимо:

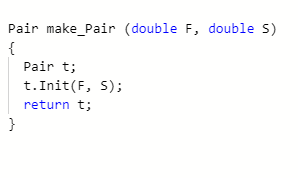
**1.1.** Создать классPair.



**1.2.** Описать методы Init, Readи Show.



**1.3.** Реализовать внешнюю функцию make\_pair().



**2.**В ходе работы были использованы следующие типы данных:

**2.1.**Double, для хранения значений длин катетов треугольника.



**3.**Для решения задачи данные были представлены в следующем виде:

**3.1.** Длины катетов треугольника были записаны в виде пары внутри класса Pair.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Операция ввода - cin.

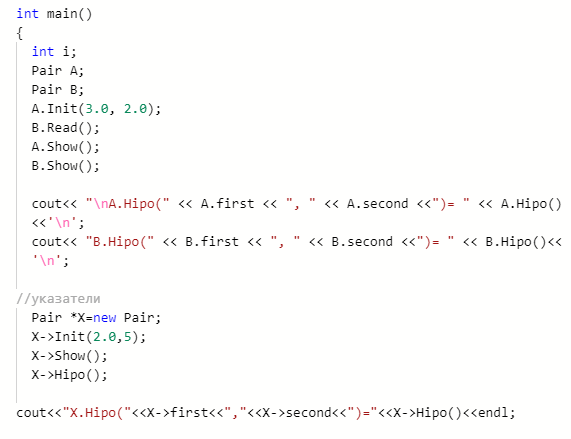


**4.2.** Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

**5.1.** Будут примены методы Init, Readи Show, а также функция make\_pair().



**Решение**

**Main()**

#include<bits/stdc++.h>

#include"pair.h"

usingnamespace std;

Pair make\_Pair (double, double);

int main()

{

int i;

Pair A;

Pair B;

A.Init(3.0, 2.0);

B.Read();

A.Show();

B.Show();

cout<<"\nA.Hipo("<< A.first <<", "<< A.second <<")= "<< A.Hipo()<<'\n';

cout<<"B.Hipo("<< B.first <<", "<< B.second <<")= "<< B.Hipo()<<'\n';

//указатели

Pair \*X=new Pair;

X->Init(2.0,5);

X->Show();

X->Hipo();

cout<<"X.Hipo("<<X->first<<","<<X->second<<")="<<X->Hipo()<<endl;

//массивы

cout<<"Работа с массивами\n";

Pair mas[3];

for (int i=0;i<3; i++) mas[i].Read();

for (i=0;i<3; i++) mas[i].Show();

for (i=0;i<3; i++)

{

mas[i].Hipo();

cout<<"mas["<<i<<"].Hipo("<<mas[i].first<<","<<mas[i].second<<")=";

cout<<mas[i].Hipo()<<endl;

}

//динамические масивы

cout<<"Работа с динамическими массивами\n";

Pair\* p\_mas=new Pair[3];//выделениепамяти

for (i=0;i<3; i++) p\_mas[i].Read();//чтениезначенийполей

for (i=0;i<3; i++) p\_mas[i].Show();//выводзначенийполей

for (i=0;i<3; i++)

{

p\_mas[i].Hipo();//вычислениестепени

cout<<"p\_mas["<<i<<"].Hipo("<<p\_mas[i].first<<","<<p\_mas[i].second; cout<<")="<<p\_mas[i].Hipo()<<endl;

}

//вызовфункции make\_Pair()

cout<<"Проверкаработы make\_Pair()\n";

double y;

double z;

cout<<"first: ";

cin>>y;

cout<<"second: ";

cin>>z;

//переменная F формируется с помощью функции make\_Pair()

Pair F=make\_Pair(y,z);

F.Show();

}

Pair make\_Pair (double F, double S)

{

Pair t;

t.Init(F, S);

return t;

}

**Pair.h**

class Pair

{

public:

double first;

double second;

void Init (double, double);

void Read();

void Show();

double Hipo();

};

**Pair.cpp**

#include<bits/stdc++.h>

#include"pair.h"

usingnamespace std;

void Pair::Init (double F, double S)

{

first=F;

second=S;

}

void Pair::Read()

{

cout<<"\nfirst: ";

cin>>first;

cout<<"\nsecond: ";

cin>>second;

}

void Pair::Show()

{

cout<<"\nfirst: "<<first;

cout<<"\nsecond: "<<second;

cout<<'\n';

}

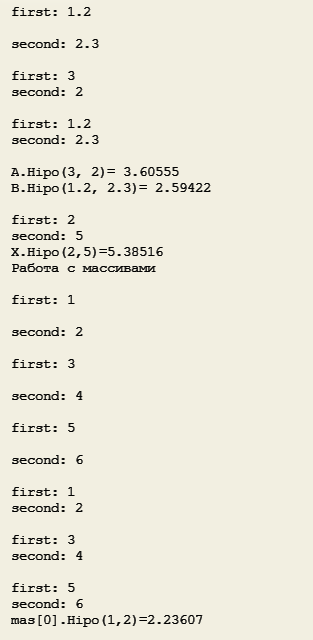
double Pair::Hipo()

{

return sqrt(first\*first+second\*second);

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Чтотакоекласс?**

Класс является абстрактным типом данных, определяемым пользователем, и представляет собой модель реального объекта в виде данных и функций дляработы с ними.

**Чтотакое объект(экземпляр)класса?**

Переменная данного класса.

**Как называютсяполякласса?**

Аттрибуты.

**Какназываютсяфункциикласса?**

Методы.

**Длячегоиспользуютсяспецификаторыдоступа?**

Для управления видимостью элементов.

**Длячегоиспользуетсяспецификаторpublic?**

Спецификатор public делает члены открытыми, доступными из любых частей кода, применяется для интерфейса класса.

**Длячегоиспользуетсяспецификаторprivate?**

Спецификатор public делает члены закрытыми, доступными лишь внутри класса.

**Если описание класса начинается со спецификатора class, то какой спецификатордоступабудетиспользоватьсяпоумолчанию?**

Private.

**Если описание класса начинается со спецификатора struct, то какой спецификатордоступабудетиспользоватьсяпоумолчанию?**

Public.

**Какой спецификатор доступа должен использоваться при описании интерфейсакласса?Почему?**

Public. Для обечения возможности взаимодействия с экземпляром класса.

**Какимобразомможноизменитьзначенияатрибутовэкземпляракласса?**

Необходимо использовать модификатор.

**Какимобразомможно получитьзначенияатрибутовэкземпляракласса?**

Необходимо использовать селектор.

**Класс описан следующим образом**

**struct Student**

**{**

**Stringname;**

**int group;**

**………**

**};**

**Объект класса определен следующим образом**

**Student \*s=new Student;**

**Как можно обратиться к полю name объекта s?**

s->name

**Класс описан следующим образом**

**struct Student**

**{**

**string name;**

**int group;**

**……..**

**};**

**Объект класса определен следующим образом**

**Student s;**

**Как можно обратиться к полю name объекта s?**

s.name

**Класс описан следующим образом**

**class Student**

**{**

**string name;**

**int group;**

**…….**

**};**

**Объект класса определен следующим образом**

**Student \*s=new Student;**

**Как можно обратиться к полю name объекта s?**

Необходимо использовать селектор.

**Класс описан следующим образом**

**class Student**

**{**

**string name;**

**int group;**

**public:**

**…..**

**};**

**Объект класса определен следующим образом**

**Student s;**

**Как можно обратиться к полю name объекта s?**

Необходимо использовать селектор.

**Класс описан следующим образом**

**class Student**

**{**

**public:**

**char\*name;**

**int group;**

**………**

**};**

**Объект класса определен следующим образом**

**Student \*s=new Student;**

**Как можно обратиться к полю name объекта s?**

s->name

## 18.2

**Постановка задачи**

(Вариант 10)

1.  Определить пользовательский класс.

2.  Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3.  Определить в классе деструктор.

4.  Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).

5.  Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.

Задание варианта:

Пользовательский класс

ЭКЗАМЕН

ФИО студента – string

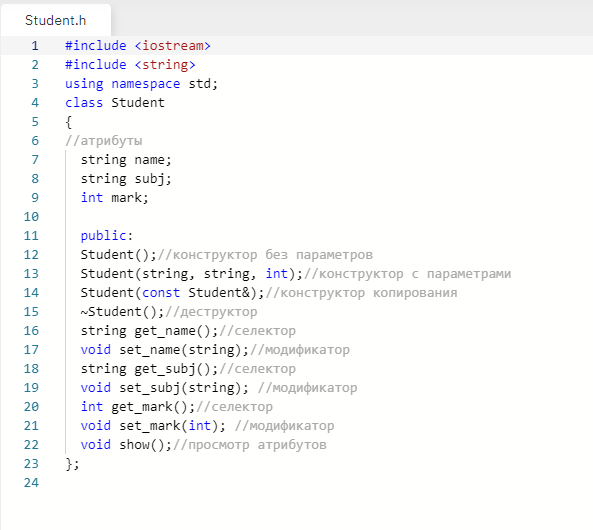
Предмет – string

Оценка – int

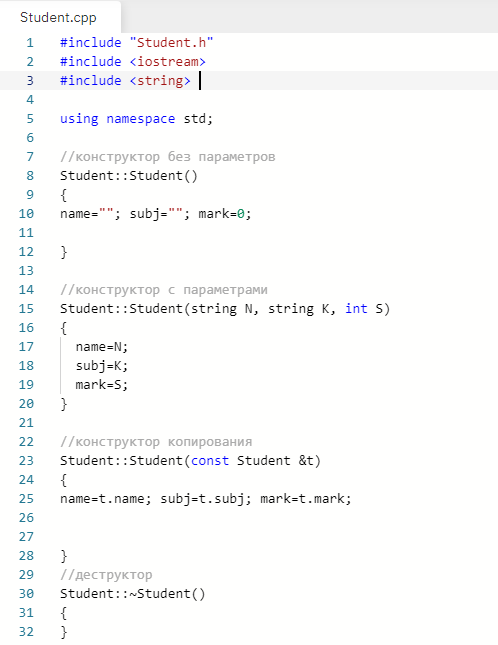
**Анализ задачи**

**1.** Для решения задачи необходимо:

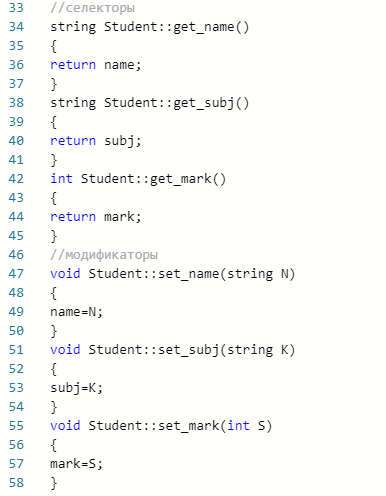
**1.1.**  Определить пользовательский класс.



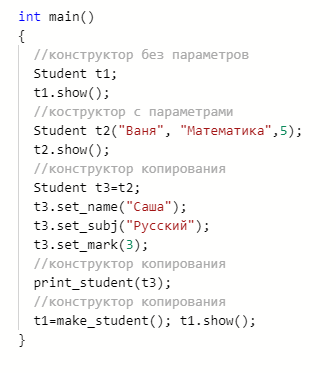
**1.2.**  Определить в классе следующие конструкторы: без параметров, с параметрами, копирования, деструктор.



**1.3.**  Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).



**1.4.**  Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.



**2.**В ходе работы были использованы следующие типы данных:

2.1.stringдля хранения ФИО студента и названия предмета.

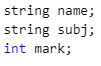


2.2. int для хранения оценки студента.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1 Данные о студентах были записаны в виде атрибутов класса student.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Операция ввода - cin.

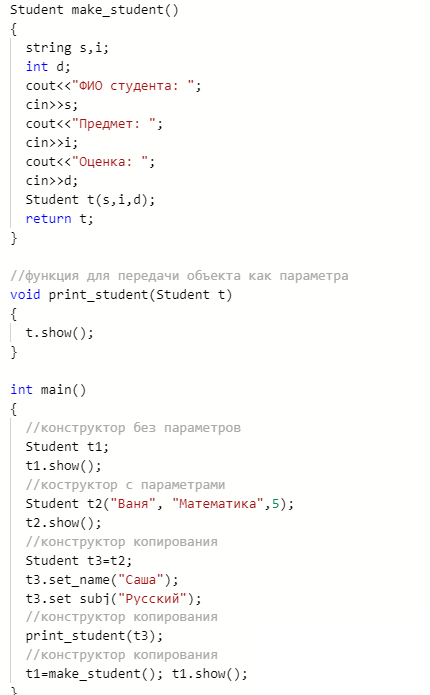


**4.2.** Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будут применены методы класса Student.



**Решение**

**Main()**

#include"Student.h"

#include<iostream>

#include<string>

usingnamespace std;

//функциядлявозвратаобъектакакрезультата

Student make\_student()

{

string s,i;

intd;

cout<<"ФИОстудента: ";

cin>>s;

cout<<"Предмет: ";

cin>>i;

cout<<"Оценка: ";

cin>>d;

Student t(s,i,d);

return t;

}

//функция для передачи объекта как параметра

void print\_student(Student t)

{

t.show();

}

int main()

{

//конструктор без параметров

Student t1;

t1.show();

//коструктор с параметрами

Student t2("Ваня", "Математика",5);

t2.show();

//конструктор копирования

Student t3=t2;

t3.set\_name("Саша");

t3.set\_subj("Русский");

t3.set\_mark(3);

//конструкторкопирования

print\_student(t3);

//конструкторкопирования

t1=make\_student(); t1.show();

}

**Student.h**

#include<iostream>

#include<string>

usingnamespace std;

class Student

{

//атрибуты

string name;

string subj;

int mark;

public:

Student();//конструктор без параметров

Student(string, string, int);//конструктор с параметрами

Student(const Student&);//конструкторкопирования

~Student();//деструктор

string get\_name();//селектор

void set\_name(string);//модификатор

string get\_subj();//селектор

void set\_subj(string); //модификатор

int get\_mark();//селектор

void set\_mark(int); //модификатор

void show();//просмотратрибутов

};

**Student.cpp**

#include"Student.h"

#include<iostream>

#include<string>

usingnamespace std;

//конструкторбезпараметров

Student::Student()

{

name=""; subj=""; mark=0;

cout<<"Конструктор без параметров: "<<this<<endl;

}

//конструкторспараметрами

Student::Student(string N, string K, int S)

{

name=N;

subj=K;

mark=S;

cout<<"Конструктор с параметрами: "<<this<<endl;

}

//конструкторкопирования

Student::Student(const Student &t)

{

name=t.name; subj=t.subj; mark=t.mark;

cout<<"Конструктор копирования: "<<this<<endl;

}

//деструктор

Student::~Student()

{

cout<<"Деструктор: "<<this<<endl;

}

//селекторы

string Student::get\_name()

{

return name;

}

string Student::get\_subj()

{

return subj;

}

int Student::get\_mark()

{

return mark;

}

//модификаторы

void Student::set\_name(string N)

{

name=N;

}

void Student::set\_subj(string K)

{

subj=K;

}

void Student::set\_mark(int S)

{

mark=S;

}

//метод для просмотра атрибутов

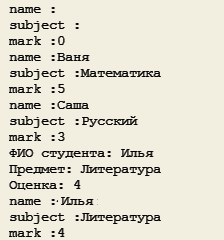
void Student::show()

{

cout<<"name :"<<name<<endl; cout<<"subject :"<<subj<<endl; cout<<"mark :"<<mark<<endl;

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Для чего нужен конструктор?**

Для автоматической инициализации переменной класса.

**Сколько типов конструкторов существует в С++?**

Три. Конструктор с параметрами, используется для инициализации объекта требуемыми значениями, Конструктор без параметров, используется для создания «пустого» объекта, Конструктор копирования, используется для создания объекта,аналогичноготому,которыйужесуществует.

**Для чего используется деструктор? В каких случаях деструктор описывается явно?**

Деструктор – это особый вид метода, применяющийся для освобождения ресурсов, выделенных конструктором объекту.Описывать в классе деструктор явным образом требуется в случае, когда объект содержит указатели на память, выделяемую динамически — иначе при уничтожении объекта память, на которую ссылались его поля-указатели, не будет помечена как свободная.

**Для чего используется конструктор без параметров? Конструктор с параметрами? Конструктор копирования?**

Конструктор с параметрами, используется для инициализации объекта требуемыми значениями, Конструктор без параметров, используется для создания «пустого» объекта, Конструктор копирования, используется для создания объекта,аналогичноготому,которыйужесуществует.

**В каких случаях вызывается конструктор копирования?**

В случае, когда есть необходимость создания объекта, аналогичного тому, который уже существует.

**Перечислить свойства конструкторов.**

Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.

Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).

Конструктор, вызываемый без параметров, называется конструктором по умолчанию.

Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию, но их может содержать только один из конструкторов.

Если программист не указал ни одного конструктора, компилятор создает его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.

Конструкторы не наследуются.

Конструкторы нельзя описывать с модификаторами const, virtual и static.

Конструкторы глобальных объектов вызываются до вызова функции main. Локальные объекты создаются, как только становится активной область их действия. Конструктор запускается и при создании временного объекта (на пример, при передаче объекта из функции).

Конструктор вызывается, если в программе встретилась какая-либо из син- таксических конструкций:

имя\_класса имя\_объекта [(список параметров)];// Список параметров не должен быть пустым

имя класса (список параметров);// Создается объект без имени (список может быть пустым)

имя\_класса имя\_объекта = выражение;// Создается объект без имени и копируется

**Перечислить свойства деструкторов.**

не имеет аргументов и возвращаемого значения;

не наследуется;

не может быть объявлен как const или static, может быть виртуальным.

**К каким атрибутам имеют доступ методы класса?**

К любым.

**Что представляет собой указатель this?**

Указатель на тот объект, для которого вызван метод, метод неявно получает в качестве аргумента этот указатель.

**Какая разница между методами определенными внутри класса и вне класса?**

Разницы нет.

**Какое значение возвращает конструктор?**

Конструктор не возвращает значений.

**Какие методы создаются по умолчанию?**

Конструктор без параметров, деструктор.

**Какое значение возвращает деструктор?**

Деструктор не возвращает значений.

**Дано описание класса class Student**

**{**

**string name;**

**int group;**

**public:**

**student(string, int);**

**student(const student&)**

**~student();**

**};**

**Какой метод отсутствует в описании класса?**

Конструктор без параметров.

**Какой метод будет вызван при выполнении следующих операторов: student\*s;**

**s=new student;**

Конструктор копирования.

**Какой метод будет вызван при выполнении следующих операторов: student s(“Ivanov”,20);**

Конструктор с параметрами.

**Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);**

**student s2=s1;**

Конструктор с параметрами.

Конструктор копирования.

**Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);**

**student s2;**

**s2=s1;**

Конструктор с параметрами.

Конструктор без параметров.

Конструктор копирования.

**Какой конструктор будет использоваться при передаче параметра в функцию**

**print(): void print(student a)**

**{**

**a.show();**

**}**

Конструктор копирования.

**Класс описан следующим образом:**

**class Student**

**{**

**string name;**

**int age;**

**public:**

**void set\_name(string);**

**void set\_age(int );**

**…..**

**};**

**Student p;**

**Каким образом можно присвоить новое значение атрибуту name объекта р?**

p.set\_name(“Сидоров”);

## 18.3

**Постановка задачи**

(Вариант 10)

1.  Определить пользовательский класс.

2.  Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3.  Определить в классе деструктор.

4.  Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).

5.  Перегрузить операцию присваивания.

6.  Перегрузить операции ввода и вывода объектов с помощью потоков.

7.  Перегрузить операции указанные в варианте.

8.  Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

Задание варианта:

Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать:

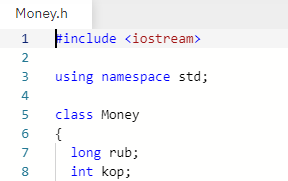
− операции сравнения (==, !=).

− вычитание копеек (--) (постфиксная и префиксная формы)

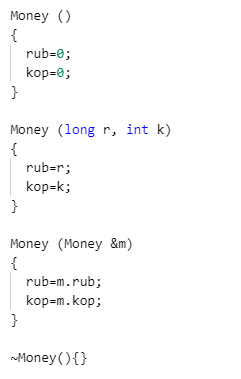
**Анализ задачи**

**1.** Для решения задачи необходимо:

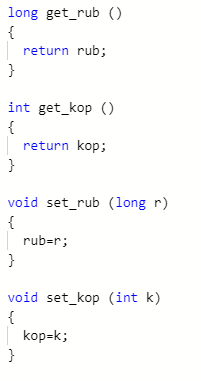
1.1.  Определить пользовательский класс.



1.2.  Определить в классе следующие конструкторы: без параметров, с параметрами, копирования, деструктор.



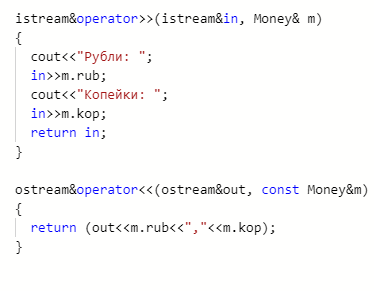
1.3.  Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).



1.4.  Перегрузить операцию присваивания.

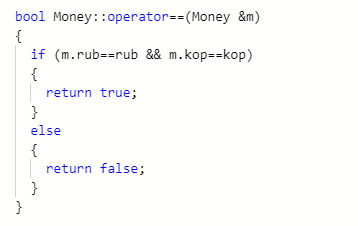


1.5.  Перегрузить операции ввода и вывода объектов с помощью потоков.

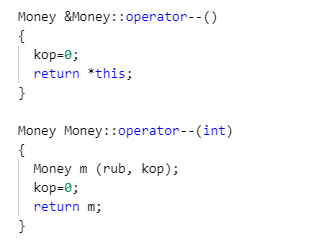


1.6.  Перегрузить операции

сравнения (==, !=)



вычитания копеек (--) (постфиксная и префиксная формы).



**2.**В ходе работы были использованы следующие типы данных:

2.1.intдля хранения количества копеек.



2.2. long для хранения количества рублей



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Данные о количестве рублей и копеек были представлены в виде атрибутов класса.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Операция ввода - cin.

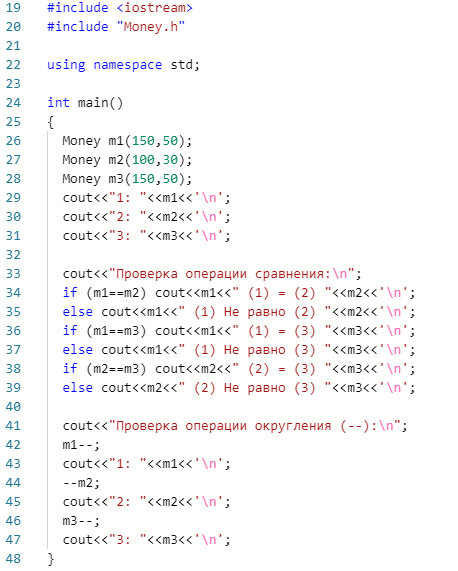


**4.2.** Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Был реализован код для демонстрации перегруженных операций.



**Решение**

**Main()**

#include<iostream>

#include"Money.h"

usingnamespace std;

int main()

{

Money m1(150,50);

Money m2(100,30);

Money m3(150,50);

cout<<"1: "<<m1<<'\n';

cout<<"2: "<<m2<<'\n';

cout<<"3: "<<m3<<'\n';

cout<<"Проверка операции сравнения:\n";

if (m1==m2) cout<<m1<<" (1) = (2) "<<m2<<'\n';

else cout<<m1<<" (1) Неравно (2) "<<m2<<'\n';

if (m1==m3) cout<<m1<<" (1) = (3) "<<m3<<'\n';

else cout<<m1<<" (1) Неравно (3) "<<m3<<'\n';

if (m2==m3) cout<<m2<<" (2) = (3) "<<m3<<'\n';

else cout<<m2<<" (2) Не равно (3) "<<m3<<'\n';

cout<<"Проверка операции округления (--):\n";

m1--;

cout<<"1: "<<m1<<'\n';

--m2;

cout<<"2: "<<m2<<'\n';

m3--;

cout<<"3: "<<m3<<'\n';

}

**Money.h**

#include<iostream>

usingnamespace std;

class Money

{

long rub;

int kop;

public:

Money ()

{

rub=0;

kop=0;

}

Money (long r, int k)

{

rub=r;

kop=k;

}

Money (Money &m)

{

rub=m.rub;

kop=m.kop;

}

~Money(){}

long get\_rub ()

{

return rub;

}

int get\_kop ()

{

return kop;

}

void set\_rub (long r)

{

rub=r;

}

void set\_kop (int k)

{

kop=k;

}

Money&operator--();

Money operator--(int);

booloperator==(Money&);

friend istream&operator>>(istream&in, Money&m);

friend ostream&operator<<(ostream&out, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

kop=0;

return \*this;

}

Money Money::operator--(int)

{

Money m (rub, kop);

kop=0;

return m;

}

bool Money::operator==(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returntrue;

}

else

{

returnfalse;

}

}

istream&operator>>(istream&in, Money& m)

{

cout<<"Рубли: ";

in>>m.rub;

cout<<"Копейки: ";

in>>m.kop;

returnin;

}

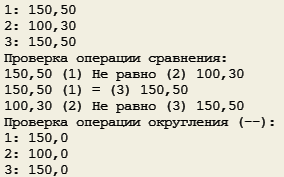
ostream&operator<<(ostream&out, const Money&m)

{

return (out<<m.rub<<","<<m.kop);

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Для чего используются дружественные функции и классы?**

Их можно использовать, если необходимо иметь доступ извне к скрытым полям класса, то есть расширить интерфейс класса.

**Сформулировать правила описания и особенности дружественных функций.**

Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ, с ключевым словом friend. В качестве параметра ей должен передаваться объект или ссылка на объект класса, поскольку указатель this ей не передается.

Дружественная функция может быть обычной функцией или методом другого ранее определенного класса. На нее не распространяется действие спецификаторов доступа, место размещения ее объявления в классе безразлично.

Одна функция может быть дружественной сразу нескольким классам.

Пример

class student;//предварительное описание класса

class teacher

{

…

void teach(student &S);

….

};

class student

{

...

friend void teacher::teach(student&);//дружественная функция,

//имеет доступ к элементам класса student

…

};

Использования дружественных функций нужно по возможности избегать, по- скольку они нарушают принцип инкапсуляции и, таким образом, затрудняют отладку и модификацию программы.

Если все методы какого-либо класса должны иметь доступ к скрытым полях другого, весь класс объявляется дружественным с помощью ключевого слова friend.

class student;//предварительное описание класса class teacher

{

…

void teach(student &S);

….

};

class student

{

...

friend class teacher;//все функции класса teacher являются

// дружественными для класса student

…

};

**Каким образом можно перегрузить унарные операции?**

Унарную операцию можно перегрузить:

Как компонентную функцию класса

Как внешнюю (глобальную) функцию

**Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?**

Один. Операнд - вызвавший ее объект.

**Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?**

Один. Операнд - параметр функции.

**Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?**

Два. Первый операнд - вызвавший ее объект, второй - параметр функции.

**Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?**

Два. Первый и второй - параметры функции.

**Чем отличается перегрузка префиксных и постфиксных унарных операций?**

Операции постфиксного инкремента и декремента должны иметь первый параметр типа int. Он используется только для того, чтобы отличить их от префиксной формы.

**Каким образом можно перегрузить операцию присваивания?**

Операция может быть перегруженна только внутри области оперделения класса. Операция-функция должна возвращать ссылку на объект, для которого она вызвана, и принимать в качестве параметра единственный аргумент - ссылку на присваиваемый объект.

**Что должна возвращать операция присваивания?**

Операция-функция должна возвращать ссылку на объект, для которого она вызвана.

**Каким образом можно перегрузить операции ввода-вывода?**

Операции ввода-вывода operator>> и operator<< всегда реализуются как внешние дружественные функции, т. к. левым операндом этих операций являются потоки.Пример:

friend istream& operator>>(istream&in, Тип&p);

friend ostream& operator<<(ostream&out, const Тип&p);

**В программе описан класс**

**class Student**

**{**

**…**

**Student& operator++();**

**….**

**};**

**и определен объект этого класса Student s;**

**Выполняется операция**

**++s;**

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Как вызов унарной функции-операции постфиксной инкрементации с одним операндом - объектом класса.

**В программе описан класс**

**class Student**

**{**

**…**

**friend Student& operator ++( Student&);**

**….**

**};**

**и определен объект этого класса Student s;**

**Выполняется операция**

**++s;**

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Как вызов дружественной унарной функции-операции постфиксной инкрементации с одним операндом - объектом класса.

**В программе описан класс class Student**

**{**

**…**

**bool operator<(Student &P);**

**….**

**};**

**и определены объекты этого класса Student a,b;**

**Выполняется операция cout<<a<b;**

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Как вызовбинарной функции-операции постфиксной инкрементации с двумя операндами - объектом класса и аргументом.

**В программе описан класс class Student**

**{**

**…**

**friend bool operator >(const Person&, Person&)**

**….**

**};**

**и определены объекты этого класса Student a,b;**

**Выполняется операция cout<<a>b;**

**Каким образом, компилятор будет воспринимать вызов функции-операции?**

Как вызов дружественной бинарной функции-операции постфиксной инкрементации с двумя операндами - объектом класса и аргументом.

## 18.4

**Постановка задачи**

(Вариант 10)

1. Определить пользовательский класс.

2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3. Определить в классе деструктор.

4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).

5. Перегрузить операцию присваивания.

6. Перегрузить операции ввода и вывода объектов с помощью потоков.

7. Определить производный класс.

8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

Задание варианта:

Базовый класс:ТРОЙКА\_ЧИСЕЛ (TRIAD)

Первое\_число (first) - int

Второе\_число (second) – int

Третье\_число (third) - int

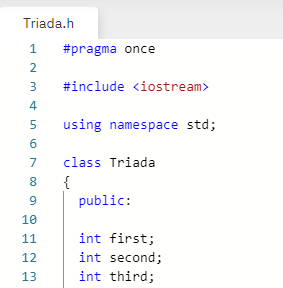
Определить методы изменения полей и увеличения полей на 1.

Создать производный класс DATE с полями год, месяц и число. Переопределить методы увеличения полей на 1 и определить метод увеличения даты на n дней.

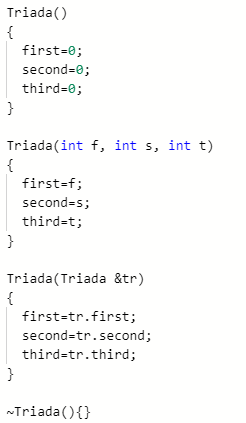
**Анализ задачи**

**1.** Для решения задачи необходимо:

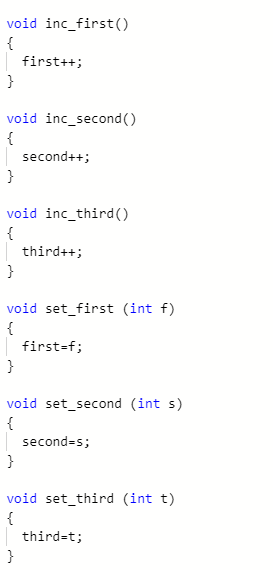
1.1. Определить пользовательский класс.



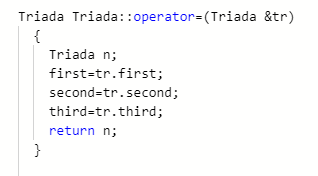
1.2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования, деструктор.



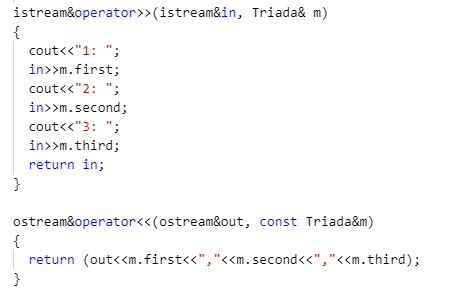
1.3. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).



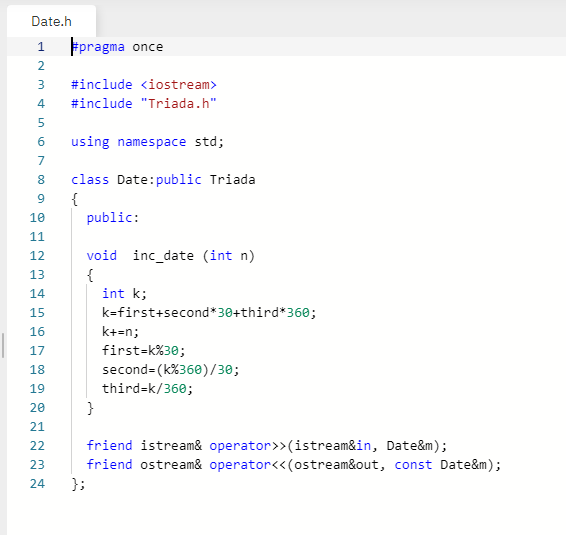
1.4. Перегрузить операцию присваивания.



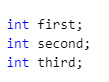
1.5. Перегрузить операции ввода и вывода объектов с помощью потоков.



1.6. Определить производный класс.

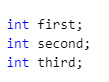


**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения информации о членах триады используется тип int

**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Для хранения информации о членах триады используются атрибуты класса.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Операция ввода - cin.



4.2. Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1.Будет реализован код, в котором продемонстрированы создание объектов и работу всех перегруженных операций.



**Решение**

**Main()**

#include"Triada.h"

#include"Date.h"

#include<iostream>

usingnamespace std;

int main()

{

Triada trd;

int a, n;

cin>>trd;

cout<<'\n'<<trd<<'\n';

// Проверка увеличения на 1

cout<<"\nУвеличение на 1\n";

trd.inc\_first();

cout<<'\n'<<trd<<'\n';

trd.inc\_second();

cout<<'\n'<<trd<<'\n';

trd.inc\_third();

cout<<'\n'<<trd<<'\n';

//

// Проверка set

cout<<"\nSet\n";

cout<<"1: ";

cin>>a;

trd.set\_first(a);

cout<<'\n'<<trd<<'\n';

cout<<"2: ";

cin>>a;

trd.set\_second(a);

cout<<'\n'<<trd<<'\n';

cout<<"3: ";

cin>>a;

trd.set\_third(a);

cout<<'\n'<<trd<<'\n';

//

// Присваивание

cout<<"\nПрисваивание\n";

Triada nTrd;

cout<<nTrd<<'\n';

nTrd=trd;

cout<<nTrd;

//

// Ввод и вывод даты

cout<<"\nВведитедату\n";

Date dt;

cin>>dt;

cout<<dt;

//

// Увеличение даты на заданное число

cout<<"\nВведите кол-во дней, на которое вы хотите увеличить дату ";

cin>>n;

dt.inc\_date(n);

cout<<dt;

//

}

**Triada.h**

#pragma once

#include<iostream>

usingnamespace std;

class Triada

{

public:

int first;

int second;

int third;

Triada()

{

first=0;

second=0;

third=0;

}

Triada(int f, int s, int t)

{

first=f;

second=s;

third=t;

}

Triada(Triada &tr)

{

first=tr.first;

second=tr.second;

third=tr.third;

}

~Triada(){}

void inc\_first()

{

first++;

}

void inc\_second()

{

second++;

}

void inc\_third()

{

third++;

}

void set\_first (int f)

{

first=f;

}

void set\_second (int s)

{

second=s;

}

void set\_third (int t)

{

third=t;

}

Triada operator=(Triada&);

friend istream&operator>>(istream&in, Triada&m);

friend ostream&operator<<(ostream&out, const Triada&m);

};

**Triada.cpp**

#include"Triada.h"

#include<iostream>

usingnamespace std;

Triada Triada::operator=(Triada &tr)

{

Triada n;

first=tr.first;

second=tr.second;

third=tr.third;

return n;

}

istream&operator>>(istream&in, Triada& m)

{

cout<<"1: ";

in>>m.first;

cout<<"2: ";

in>>m.second;

cout<<"3: ";

in>>m.third;

returnin;

}

ostream&operator<<(ostream&out, const Triada&m)

{

return (out<<m.first<<","<<m.second<<","<<m.third);

}

**Date.h**

#pragma once

#include<iostream>

#include"Triada.h"

usingnamespace std;

class Date:public Triada

{

public:

void inc\_date (int n)

{

int k;

k=first+second\*30+third\*360;

k+=n;

first=k%30;

second=(k%360)/30;

third=k/360;

}

friend istream&operator>>(istream&in, Date&m);

friend ostream&operator<<(ostream&out, const Date&m);

};

**Date.cpp**

#include"Date.h"

#include<iostream>

usingnamespace std;

istream&operator>>(istream&in, Date& m)

{

cout<<"Число: ";

in>>m.first;

cout<<"Месяц: ";

in>>m.second;

cout<<"Год: ";

in>>m.third;

returnin;

}

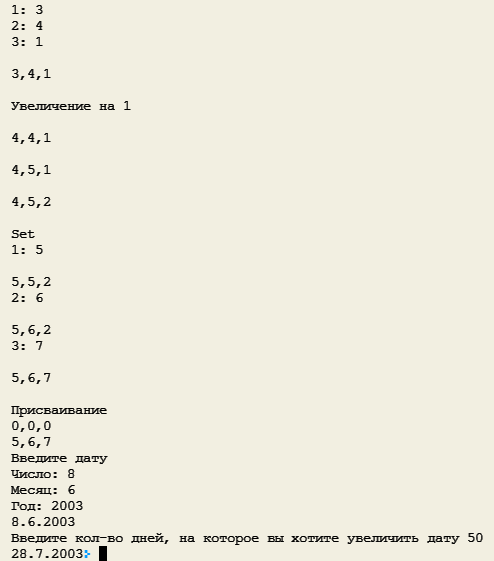
ostream&operator<<(ostream&out, const Date&m)

{

return (out<<m.first<<"."<<m.second<<"."<<m.third);

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Для чего используется механизм наследования?**

Для получения нового класса на основе уже существующего. Существующий класс может быть дополнен или изменен для создания нового класса.

**Каким образом наследуются компоненты класса, описанные со спецификатором public?**

Член класса может использоваться любой функцией, которая является членом данного или производного класса, а также возможен доступ извне через имя объекта.

**Каким образом наследуются компоненты класса, описанные со спецификатором private?**

Член класса может использоваться только функциями – членами данного класса и функциями – “друзьями” своего класса. В производном классе он недоступен.

**Каким образом наследуются компоненты класса, описанные со спецификатором protected?**

Член класса может использоваться функциями – членами данного и производных классов и функциями – “друзьями” своего и производных классов.

**Каким образом описывается производный класс?**

При описании через двоеточие указывается спецификатор доступа для назначения его наследуемым объектам, а также название класса-родителя.

Пример:

class имяКласса : publicсписокБазовыхКлассов

{списокКомпонентовКласса};

**Наследуются ли конструкторы?**

Нет.

**Наследуются ли деструкторы?**

Нет.

**В каком порядке конструируются объекты производных классов?**

Объекты класса конструируются снизу вверх: сначала базовый, потом компоненты- объекты (если они имеются), а потом сам производный класс. Таким образом, объект производного класса содержит в качестве подобъекта объект базового класса.

**В каком порядке уничтожаются объекты производных классов?**

Уничтожаются объекты в следующем порядке: сначала производный, потом его компоненты-объекты, а потом базовый объект.Таким образом, порядок уничтожения объекта противоположен по отношению к порядку его конструирования.

**Что представляют собой виртуальные функции и механизм позднего связывания?**

К механизму виртуальных функций обращаются в тех случаях, когда в каждом производном классе требуется свой вариант некоторой компонентной функции. Классы, включающие такие функции, называются полиморфными.Виртуальные функции предоставляют механизм позднего (отложенного) или динамического связывания. Любая нестатическая функция базового класса может быть сделана виртуальной, для чего используется ключевое слово virtual. Таким образом, интерпретация каждого вызова виртуальной функции через указатель на базовый класс зависит от значения этого указателя, т.е. от типа объекта, для которого выполняется вызов. Выбор того, какую виртуальную функцию вызвать, будет зависеть от типа объекта, на который фактически (в момент выполнения программы) направлен указатель, а не от типа указателя.

Для выполнения позднего связывания используются указатели на функции. Указатель на функцию — это тип указателя, который указывает на функцию вместо переменной. Функция, на которую указывает указатель, может быть вызвана через указатель и оператор вызова функции.

**Могут ли быть виртуальными конструкторы? Деструкторы?**

Конструкторы не могут быть виртуальными.

Деструкторы могут.

**Наследуется ли спецификатор virtual?**

Да. После того как функция определена как виртуальная, ее повторное определение в производном классе (с тем же самым прототипом) создает в этом классе новую виртуальную функцию, причем спецификатор virtual может не использоваться.

**Какое отношение устанавливает между классами открытое наследование?**

Открытое наследование устанавливает между классами отношение «является»: класс-наследник является частью класса-родителя. Данное положение называется принципом подстановки.

**Какое отношение устанавливает между классами закрытое наследование?**

Закрытое наследование – это наследование реализации, в этом случае принцип подстановки не соблюдается.

**В чем заключается принцип подстановки?**

Принцип звучит так: везде, где может быть использован объект базового класса (при присваивании, при передаче параметров и возврате результата), вместо него разрешается использовать объект производного класса.

**Имеется иерархия классов:**

**class Student**

**{**

**public:**

**int age;**

**string name;**

**...**

**};**

**class Employee : public Student**

**{**

**protected:**

**string post;**

**...**

**};**

**class Teacher : public Employee**

**{**

**protected: int stage;**

**...**

**};**

**Teacher x;**

**Какие компонентные данные будет иметь объект х?**

string name;

string post;

int stage;

**Для классов Student, Employee и Teacher написать конструкторы без параметров.**

Student()

{

age=0;

name=””;

}

Employee()

{

name=””;

post=””;

}

Teacher()

{

name=””;

post=””;

stage=0;

}

**Для классов Student, Employee и Teacher написать конструкторы с параметрами.**

Student(int a, string n)

{

age=a;

name=n;

}

Employee(string n, string p)

{

name=n;

post=p;

}

Teacher(string n, string p, int s)

{

name=n;

post=p;

stage=s;

}

**Для классов Student, Employee и Teacher написать конструкторы копирования.**

Student(Student& s)

{

age=s.age;

name=s.name;

}

Employee(Employee& e)

{

name=e.name;

post=e.post;

}

Teacher(Teacher& t)

{

name=t.name;

post=t.post;

stage=t.stage;

}

**Для классов Student, Employee и Teacher определить операцию присваивания.**

Student&Student::operator =(Student& s)

{

this->age=s.age;

this->name=s.name;

}

Employee&Employee::operator =((Employee& e)

{

this->name=e.name;

this->post=e.post;

}

Teacher&Teacher::operator =((Teacher& t)

{

this->name=t.name;

this->post=t.post;

this->stage=t.stage;

}

## 18.5

**Постановка задачи**

(Вариант 10)

1.  Определить абстрактный класс.

2.  Определить иерархию классов, в основе которой будет находиться абстрактный класс (см. лабораторную работу №4).

3.  Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.

4.  Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.

5.  В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

Задание варианта:

Базовый класс: ТРОЙКА\_ЧИСЕЛ (TRIADA)

Первое\_число (first) - int

Второе\_число (second) – int

Третье\_число (third) - int

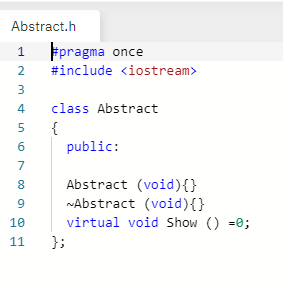
Определить методы изменения полей и увеличения полей на 1.

Создать производный класс DATE с полями год, месяц и число. Переопределить методы увеличения полей на 1 и определить метод увеличения даты на n дней.

**Анализ задачи**

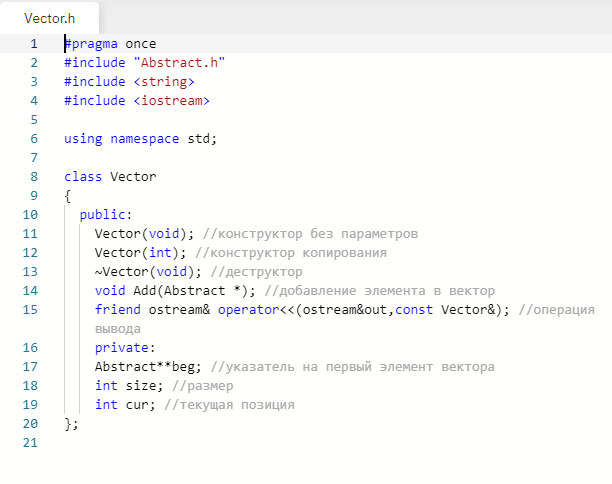
**1.** Для решения задачи необходимо:

1.1.  Определить абстрактный класс.

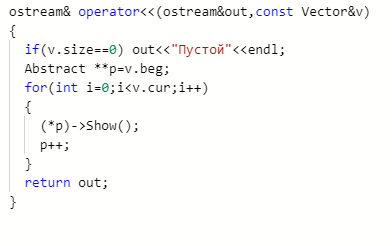


1.2.  Определить иерархию классов, в основе которой будет находиться абстрактный класс

1.3.  Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.

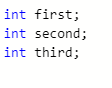


1.4.  Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.



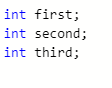
**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения информации об элементах вектора используется тип int.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Для хранения информации об элементах вектора, эти элементы представлены в виде атрибутов класса.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Операция ввода - cin.



4.2. Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, в котором продемонстрированы перегруженные операции и полиморфизм Вектора.



**Решение**

**Main()**

#include"Triada.h"

#include"Date.h"

#include"Vector.h"

#include<iostream>

usingnamespace std;

int main()

{

Abstract \*p;

Vector v(5);

Triada trd;

int a, n;

cin>>trd;

cout<<'\n'<<trd<<'\n';

// Проверка увеличения на 1

cout<<"\nУвеличение на 1\n";

trd.inc\_first();

cout<<'\n'<<trd<<'\n';

trd.inc\_second();

cout<<'\n'<<trd<<'\n';

trd.inc\_third();

cout<<'\n'<<trd<<'\n';

//

// Проверка set

cout<<"\nSet\n";

cout<<"1: ";

cin>>a;

trd.set\_first(a);

cout<<'\n'<<trd<<'\n';

cout<<"2: ";

cin>>a;

trd.set\_second(a);

cout<<'\n'<<trd<<'\n';

cout<<"3: ";

cin>>a;

trd.set\_third(a);

cout<<'\n'<<trd<<'\n';

//

// Присваивание

cout<<"\nПрисваивание\n";

Triada nTrd;

cout<<nTrd<<'\n';

nTrd=trd;

cout<<nTrd;

//

// Ввод и вывод даты

cout<<"\nВведитедату\n";

Date dt;

cin>>dt;

cout<<dt;

//

// Увеличение даты на заданное число

cout<<"\nВведите кол-во дней, на которое вы хотите увеличить дату ";

cin>>n;

dt.inc\_date(n);

cout<<dt;

//

// Работа с вектором

Triada c;

cout<<"\nКласс Triada\n";

cin>>c;

Date b;

cout<<"Класс Date\n";

cin>>b;

p=&c;

v.Add(p);

p=&b;

v.Add(p);

cout<<"Вектор\n";

cout<<v;

//

}

**Abstract.h**

#pragma once

#include<iostream>

class Abstract

{

public:

Abstract (void){}

~Abstract (void){}

virtualvoid Show () =0;

};

**Vector.h**

#pragma once

#include"Abstract.h"

#include<string>

#include<iostream>

usingnamespace std;

class Vector

{

public:

Vector(void); //конструкторбезпараметров

Vector(int); //конструктор копирования

~Vector(void); //деструктор

void Add(Abstract \*); //добавление элемента в вектор

friend ostream&operator<<(ostream&out,const Vector&); //операциявывода

private:

Abstract\*\*beg; //указатель на первый элемент вектора

int size; //размер

int cur; //текущая позиция

};

**Vector.cpp**

#include"Vector.h"

usingnamespacestd;

//конструкторбезпараметров

Vector::Vector(void)

{

beg=0;

size=0;

cur=0;

}

//деструктор

Vector::~Vector(void)

{

if(beg!=0)delete [] beg; beg=0;

}

//конструктор с параметрами

Vector::Vector(int n)

{

beg=new Abstract\*[n]; cur=0;

size=n;

}

//добавление объекта, на который указывает указатель p в вектор

void Vector::Add(Abstract \*p)

{

if(cur<size)

{

beg[cur]=p; cur++;

}

}

//операциявывода

ostream&operator<<(ostream&out,const Vector&v)

{

if(v.size==0) out<<"Пустой"<<endl;

Abstract \*\*p=v.beg;

for(int i=0;i<v.cur;i++)

{

(\*p)->Show();

p++;

}

return out;

}

**Triada.h**

#pragma once

#include<iostream>

#include"Abstract.h"

usingnamespace std;

class Triada: public Abstract

{

public:

int first;

int second;

int third;

Triada()

{

first=0;

second=0;

third=0;

}

Triada(int f, int s, int t)

{

first=f;

second=s;

third=t;

}

Triada(Triada &tr)

{

first=tr.first;

second=tr.second;

third=tr.third;

}

virtual ~Triada(){}

void inc\_first()

{

first++;

}

void inc\_second()

{

second++;

}

void inc\_third()

{

third++;

}

void set\_first (int f)

{

first=f;

}

void set\_second (int s)

{

second=s;

}

void set\_third (int t)

{

third=t;

}

void Show ()

{

cout<<\*this<<'\n';

}

Triada operator=(Triada&);

friend istream&operator>>(istream&in, Triada&m);

friend ostream&operator<<(ostream&out, const Triada&m);

};

**Triada.cpp**

#include"Triada.h"

#include<iostream>

usingnamespace std;

Triada Triada::operator=(Triada &tr)

{

Triada n;

first=tr.first;

second=tr.second;

third=tr.third;

return n;

}

istream&operator>>(istream&in, Triada& m)

{

cout<<"1: ";

in>>m.first;

cout<<"2: ";

in>>m.second;

cout<<"3: ";

in>>m.third;

returnin;

}

ostream&operator<<(ostream&out, const Triada&m)

{

return (out<<m.first<<","<<m.second<<","<<m.third);

}

**Date.h**

#pragma once

#include<iostream>

#include"Triada.h"

usingnamespace std;

class Date:public Triada

{

public:

void inc\_date (int n)

{

int k;

k=first+second\*30+third\*360;

k+=n;

first=k%30;

second=(k%360)/30;

third=k/360;

}

void Show ()

{

cout<<first<<"."<<second<<"."<<third<<'\n';

}

friend istream&operator>>(istream&in, Date&m);

friend ostream&operator<<(ostream&out, const Date&m);

};

**Date.cpp**

#include"Date.h"

#include<iostream>

usingnamespace std;

istream&operator>>(istream&in, Date& m)

{

cout<<"Число: ";

in>>m.first;

cout<<"Месяц: ";

in>>m.second;

cout<<"Год: ";

in>>m.third;

returnin;

}

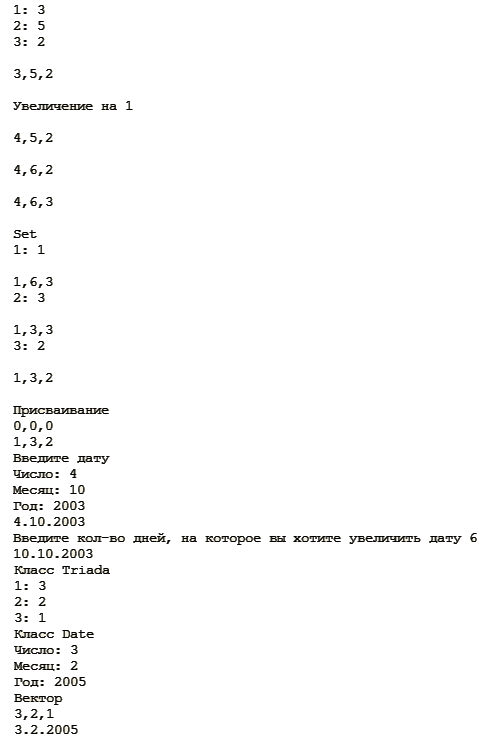
ostream&operator<<(ostream&out, const Date&m)

{

return (out<<m.first<<"."<<m.second<<"."<<m.third);

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?**

Чисто виртуальный метод содержит признак = 0 вместо тела, например:

virtual void f(int) = 0;

Чисто виртуальный метод должен переопределяться в производном классе (возможно, опять как чисто виртуальный).

**Какой класс называется абстрактным?**

Класс, содержащий хотя бы один чисто виртуальный метод.

**Для чего предназначены абстрактные классы?**

Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах. Абстрактный класс может использоваться только в качестве базового для других классов — объекты абстрактного класса создавать нельзя, поскольку прямой или косвенный вызов чисто виртуального метода приводит к ошибке при выполнении.

**Что такое полиморфные функции?**

Функция, параметром которой является указатель на абстрактный класс. На место этого параметра при выполнении программы может передаваться указатель на объект любого производного класса. Это позволяет работать с объектом любого типа в пределах одной иерархии.

**Чем полиморфизм отличается от принципа подстановки?**

*Цитата из инета: тут сравнивается кислое и квадратное.*

*Поставил определения…*

Принцип звучит так: везде, где может быть использован объект базового класса (при присваивании, при передаче параметров и возврате результата), вместо него разрешается использовать объект производного класса.

Полиморфизм- способность функции работать с переменными разных классов одной иерархии благодаря наличию в основании иерархии абстрактного класса.

**Привести примеры иерархий с использованием абстрактных классов.**

**Abstract.h**

#pragma once

#include<iostream>

class Abstract

{

public:

Abstract (void){}

~Abstract (void){}

Virtual void Show () =0;

};

**Triada.h**

#pragma once

#include<iostream>

#include"Abstract.h"

usingnamespace std;

class Triada: public Abstract

{

public:

int first;

int second;

int third;

Triada()

{

first=0;

second=0;

third=0;

}

Triada(int f, int s, int t)

{

first=f;

second=s;

third=t;

}

Triada(Triada &tr)

{

first=tr.first;

second=tr.second;

third=tr.third;

}

virtual ~Triada(){}

void inc\_first()

{

first++;

}

void inc\_second()

{

second++;

}

void inc\_third()

{

third++;

}

void set\_first (int f)

{

first=f;

}

void set\_second (int s)

{

second=s;

}

void set\_third (int t)

{

third=t;

}

void Show ()

{

cout<<\*this<<'\n';

}

Triada operator=(Triada&);

friend istream&operator>>(istream&in, Triada&m);

friend ostream&operator<<(ostream&out, const Triada&m);

};

**Date.h**

#pragma once

#include<iostream>

#include"Triada.h"

usingnamespace std;

class Date:public Triada

{

public:

void inc\_date (int n)

{

int k;

k=first+second\*30+third\*360;

k+=n;

first=k%30;

second=(k%360)/30;

third=k/360;

}

void Show ()

{

cout<<first<<"."<<second<<"."<<third<<'\n';

}

friend istream&operator>>(istream&in, Date&m);

friend ostream&operator<<(ostream&out, const Date&m);

};

**Привести примеры полиморфных функций.**

virtual void Show () =0;

void Show ()

{

cout<<\*this<<'\n';

}

void Show ()

{

cout<<first<<"."<<second<<"."<<third<<'\n';

}

**В каких случаях используется механизм позднего связывания?**

Для выполнения позднего связывания используются указатели на функции. Указатель на функцию — это тип указателя, который указывает на функцию вместо переменной. Функция, на которую указывает указатель, может быть вызвана через указатель и оператор вызова функции.

Он используется в случаях, когда заранее неизвестно, какой вариант полиморфной функции будет вызван.

## 18.6

**Постановка задачи**

(Вариант 10)

1.  Определить класс-контейнер.

2.  Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

3.  Перегрузить операции.

4.  Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.

Класс- контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

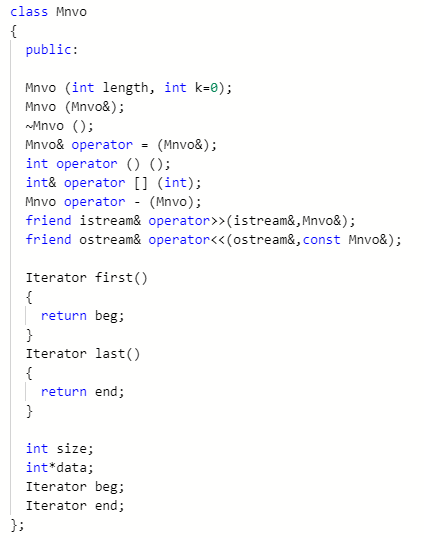
() – определение размера вектора;

- – разность множеств;

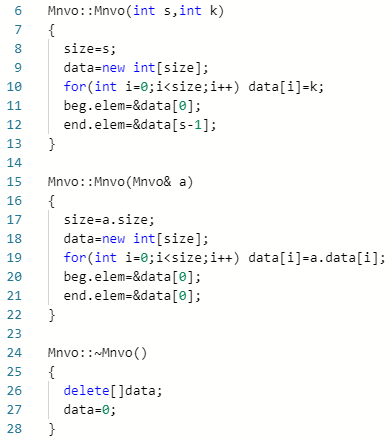
**Анализ задачи**

1. Для решения задачи необходимо:

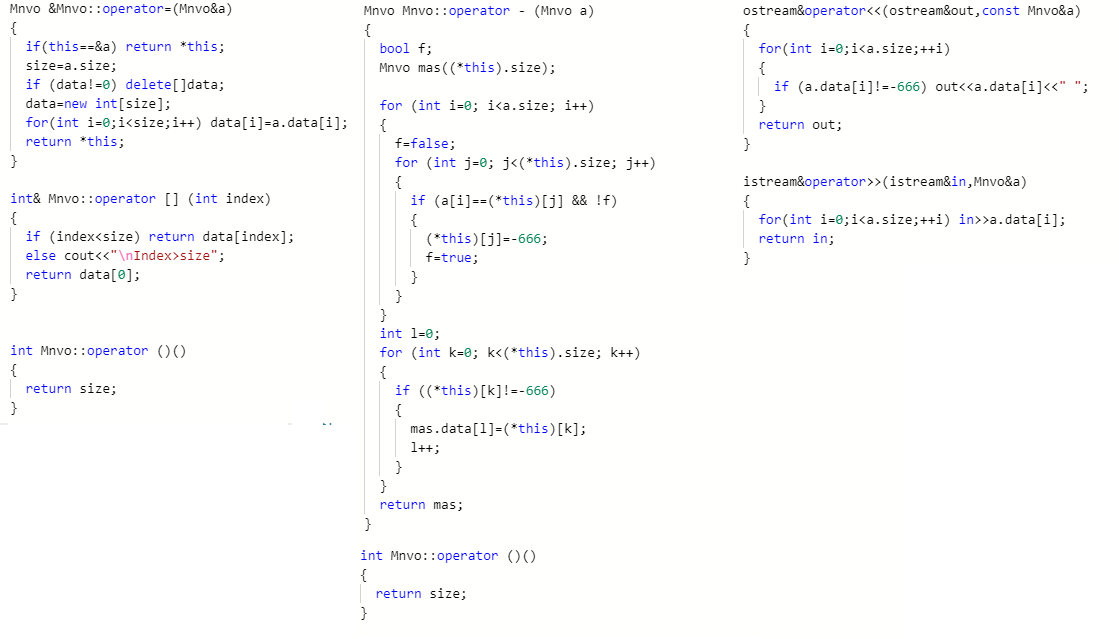
1.1.  Определить класс-контейнер.



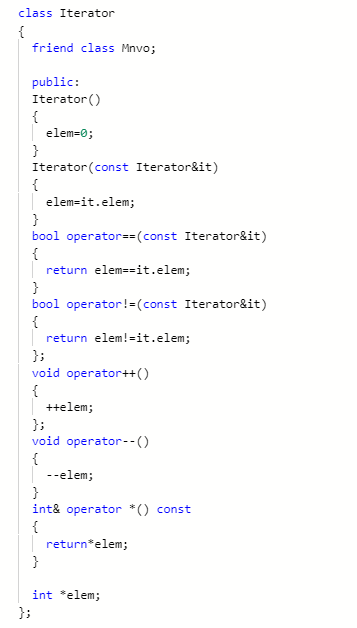
1.2.  Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.



1.3.  Перегрузить операции.



1.4.  Реализовать класс-итератор.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов используется тип int.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Данные были представлены в виде атрибутов класса Mnvo



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Операция ввода - cin.



4.2. Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет написан код, проверяющий работу перегруженных функций.



**Решение**

**intmain()**

#include"Mnvo.h"

#include<iostream>

usingnamespace std;

void make\_mnvo(Mnvo&, int);

int main()

{

Mnvo a(5);

cout<<"Нулевоемножество\n"<<a<<"\n";

cout<<"Введите ваше множество\n";

cin>>a[0];

make\_mnvo(a, 5);

cout<<a<<"\n";

cout<<"Проверка операции =\n";

a[0]=100;

cout<<a<<"\n";

Mnvo b(10);

cout<<"Нулевое множество\n"<<b<<"\n";

b=a;

cout<<"множество b после применения операции =\n"<<b<<"\n";

cout<<"Проверка операции ()\nдлина a="<<a()<<endl;

cout<<"Вывод элементов через итератор\n";

Iterator i=a.first();

cout<<"Первыйэлемент= "<<\*i<<endl;

++i;

cout<<"Второй элемент= "<<\*i<<endl;

cout<<"Вывод всего множеста через итератор\n";

for (i=a.first(); i!=a.last(); ++i) cout<<\*i<<endl;

cout<<\*i<<endl;

Mnvo n(3);

cout<<"Проверка операции - для множеств\n";

cout<<"Введите множество, которое вы хотите вычесть (длина- 3)\n";

cin>>n;

a-n;

cout<<a<<"\n";

}

void make\_mnvo(Mnvo& A, int length)

{

int i,j,el;

for (i = 1; i < length; i++)

  {

    cin >> el;

    if (el > A[i-1])

    {

      A[i] = el;

    }

    else

    {

      j = i-1;

      while (el < A[j])

      {

        A[j + 1] = A[j];

        A[j] = 0;

        j--;

     }

      A[j + 1] = el;

    }

  }

}

**Mnvo.h**

#include<iostream>

usingnamespace std;

class Iterator

{

friendclass Mnvo;

public:

Iterator()

{

elem=0;

}

Iterator(const Iterator&it)

{

elem=it.elem;

}

booloperator==(const Iterator&it)

{

return elem==it.elem;

}

booloperator!=(const Iterator&it)

{

return elem!=it.elem;

};

voidoperator++()

{

++elem;

};

voidoperator--()

{

--elem;

}

int&operator \*() const

{

return\*elem;

}

int \*elem;

};

class Mnvo

{

public:

Mnvo (int length, int k=0);

Mnvo (Mnvo&);

~Mnvo ();

Mnvo&operator = (Mnvo&);

intoperator () ();

int&operator [] (int);

Mnvo operator - (Mnvo);

friend istream&operator>>(istream&,Mnvo&);

friend ostream&operator<<(ostream&,const Mnvo&);

Iterator first()

{

return beg;

}

Iterator last()

{

return end;

}

int size;

int\*data;

Iterator beg;

Iterator end;

};

**Mnvo.cpp**

#include<iostream>

#include"Mnvo.h"

usingnamespace std;

Mnvo::Mnvo(int s,int k)

{

size=s;

data=newint[size];

for(int i=0;i<size;i++) data[i]=k;

beg.elem=&data[0];

end.elem=&data[s-1];

}

Mnvo::Mnvo(Mnvo& a)

{

size=a.size;

data=newint[size];

for(int i=0;i<size;i++) data[i]=a.data[i];

beg.elem=&data[0];

end.elem=&data[0];

}

Mnvo::~Mnvo()

{

delete[]data;

data=0;

}

Mnvo &Mnvo::operator=(Mnvo&a)

{

if(this==&a) return \*this;

size=a.size;

if (data!=0) delete[]data;

data=newint[size];

for(int i=0;i<size;i++) data[i]=a.data[i];

return \*this;

}

int& Mnvo::operator [] (int index)

{

if (index<size) return data[index];

else cout<<"\nIndex>size";

return data[0];

}

int Mnvo::operator ()()

{

return size;

}

Mnvo Mnvo::operator - (Mnvo a)

{

bool f;

Mnvo mas((\*this).size);

for (int i=0; i<a.size; i++)

{

f=false;

for (int j=0; j<(\*this).size; j++)

{

if (a[i]==(\*this)[j] && !f)

{

(\*this)[j]=-666;

f=true;

}

}

}

int l=0;

for (int k=0; k<(\*this).size; k++)

{

if ((\*this)[k]!=-666)

{

mas.data[l]=(\*this)[k];

l++;

}

}

return mas;

}

ostream&operator<<(ostream&out,const Mnvo&a)

{

for(int i=0;i<a.size;++i)

{

if (a.data[i]!=-666) out<<a.data[i]<<" ";

}

return out;

}

istream&operator>>(istream&in,Mnvo&a)

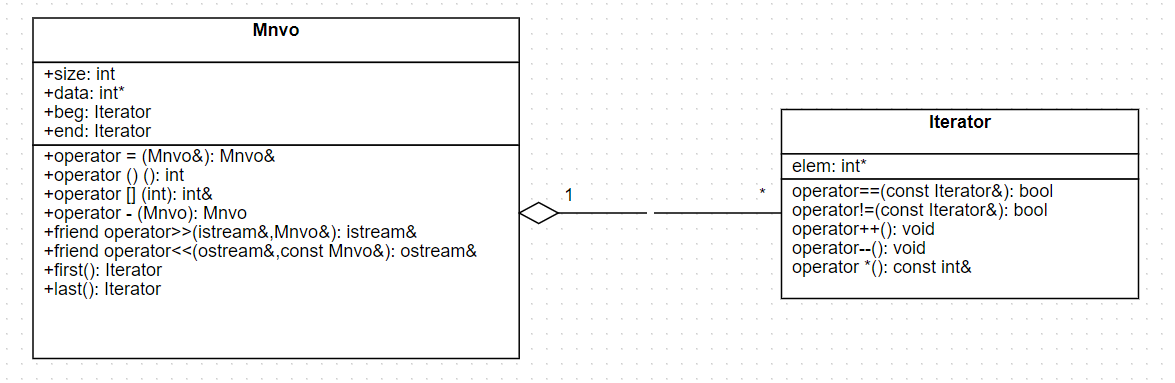
{

for(int i=0;i<a.size;++i) in>>a.data[i];

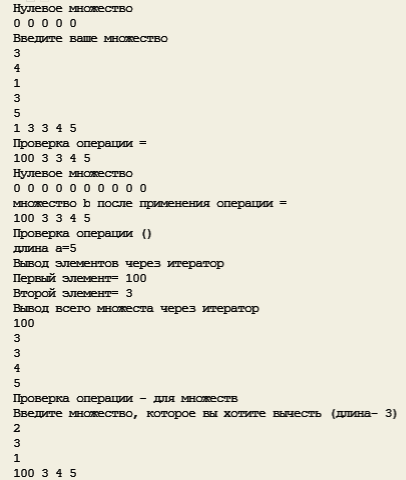
return in;

}

**Диаграмма классов**

****

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Что такое абстрактный тип данных? Привести примеры АТД.**

АТД - тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

Примеры: Класс, Array, List, Map, Queue, Stack, Table, Tree, Vector

**Привести примеры абстракции через параметризацию.**

#include <stdio.h>

struct date

{

int month; // месяц

int day; // день

int year; // год

};

void set\_date(date\* f, int d, int m, int y)

{

f->day = d;

f->month = m;

f->year = y;

}

void print\_date(date\* f)

{

printf("%d.%d.%d", f->day, f->month, f->year);

}

int main()

{

date today;

set\_date(&today, 2, 4, 2014);

print\_date(&today);

getchar();

return 0;

}

**Привести примеры абстракции через спецификацию.**

#include <stdio.h>

struct date

{

int month; // месяц

int day; // день

int year; // год

};

void set\_date(date\* f, int d, int m, int y)

{

f->day = d;

f->month = m;

f->year = y;

}

void print\_date(date\* f)

{

printf("%d.%d.%d", f->day, f->month, f->year);

}

int main()

{

date today;

set\_date(&today, 2, 4, 2014);

print\_date(&today);

getchar();

return 0;

}

**Что такое контейнер? Привести примеры.**

Контейнер – набор однотипных элементов. Пример: Встроенный массив.

**Какие группы операций выделяют в контейнерах?**

Операции доступа к элементам, которые обеспечивают и операцию замены значений элементов

Операции добавления и удаления элементов или групп элементов

Операции поиска элементов и групп элементов

Операции объединения контейнеров

Специальные операции, которые зависят от вида контейнера

**Какие виды доступа к элементам контейнера существуют? Привести примеры.**

Последовательный (список, через указатель на следующий элемент), прямой (массив, через индекс) и ассоциативный (словарь, через значение парной переменной).

**Что такое итератор?**

Итератор – это объект, который обеспечивает последовательный доступ к элементам контейнера.

**Каким образом может быть реализован итератор?**

Итератор может быть реализован как часть класса-контейнера в виде набора методов.

v.first() перейти к первому элементу

v.last() перейти к последнему элементу

v.next() перейти к следующему элементу

v.prev() перейти к предыдущему элементу

v.skip(n) перейти на n элементов вперед

v.skip(-n) перейти на n элементов назад

v.current() получить текущий элемент

**Каким образом можно организовать объединение контейнеров?**

Простое сцепление двух контейнеров: в новый контейнер попадают сначала элементы первого контейнера, потом второго, операция не коммутативна.

Объединение упорядоченных контейнеров, новый контейнер тоже будет упорядочен, операция коммутативна.

Объединение контейнеров как объединение множеств, в новый контейнер попадают только те элементы, которые есть хотя бы в одном контейнере, операция коммутативна.

Объединение контейнеров как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна.

Для контейнеров-множеств может быть еще реализована операция вычитания, в контейнер попадают только те элементы первого контейнера, которых нет во втором, операция не коммутативна.

Извлечение части элементов из контейнера и создание нового контейнера. Эта операция может быть выполнена с помощью конструктора, а часть контейнера задается двумя итераторами.

**Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?**

Ассоциативный.

**Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?**

Стек.

**Какой из объектов (a,b,c,d) является контейнером?**

**a. int mas=10;**

**b. int mas;**

**c. struct {char name[30]; int age;} mas;**

**d. int mas[100];**

D.

**Какой из объектов (a,b,c,d) не является контейнером?**

**a. int a[]={1,2,3,4,5};**

**b. int mas[30];**

**c. struct {char name[30]; int age;} mas[30];**

**d. int mas;**

D.

**Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?**

Прямой.

**Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?**

Последовательный.

## 18.7

**Постановка задачи**

(Вариант 10)

1.  Определить шаблон класса-контейнера.

2.  Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

3.  Перегрузить операции, указанные в варианте.

4.  Инстанцировать шаблон для стандартных типов данных (int, float, double).

5.  Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы стандартных типов данных.

6.  Реализовать пользовательский класс.

7.  Перегрузить для пользовательского класса операции ввода-вывода.

8.  Перегрузить операции необходимые для выполнения операций контейнерного класса.

9.  Инстанцировать шаблон для пользовательского класса.

10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.

Класс- контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора;

- – разность множеств;

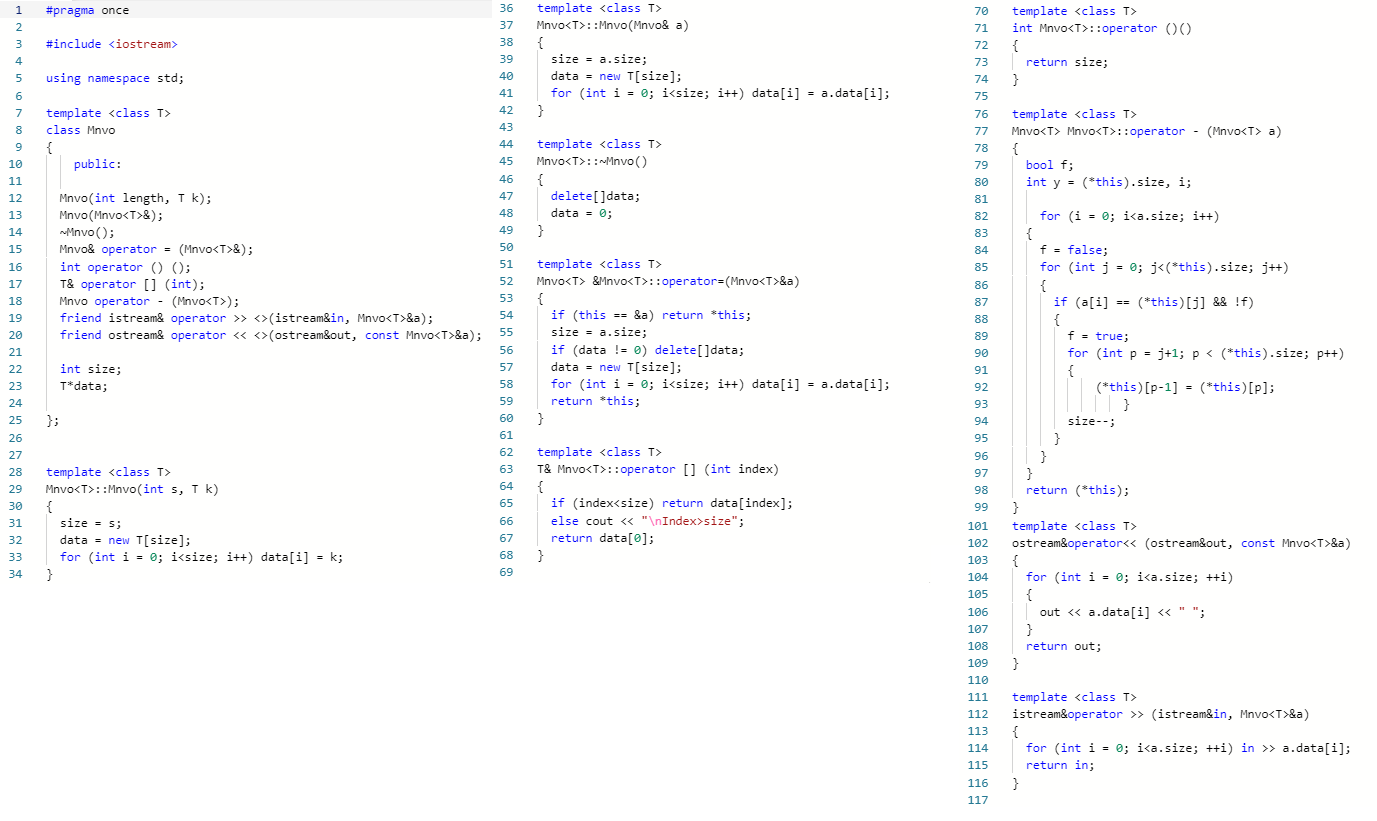
Пользовательский класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек.

Дробная часть числа при выводе на экран должна быть отделена от целой части запятой.

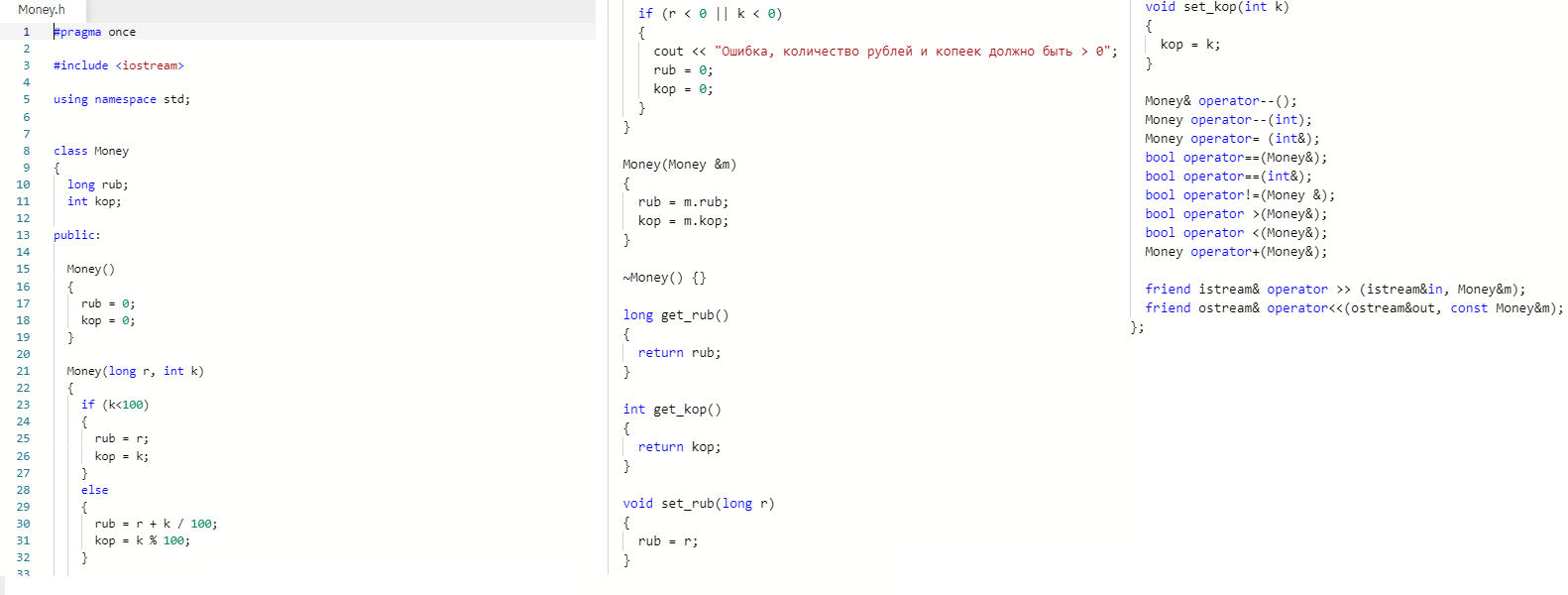
**Анализ задачи**

1. Для решения задачи необходимо:

1.1.  Определить шаблон класса-контейнера. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания. Перегрузить операции.



1.2.  Реализовать пользовательский класс.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов – тип int.



2.2. Для хранения значений элементов – тип Money.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Данные хранятся в классе Mnvo.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Операция ввода - cin.

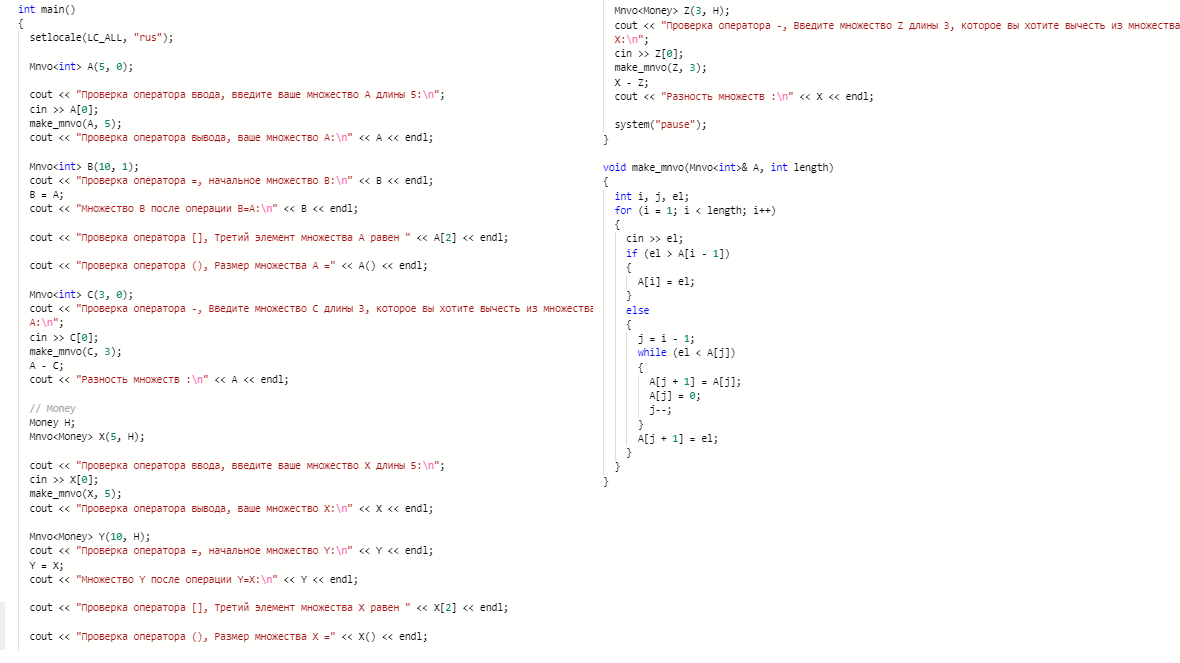


4.2. Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код для демонстрации работы параметризированного класса.



**Решение**

**Main.cpp**

#include<iostream>

#include"Mnvo.h"

#include"Money.h"

usingnamespace std;

void make\_mnvo(Mnvo<int>& A, int length);

void make\_mnvo(Mnvo<Money>& A, int length);

int main()

{

  setlocale(LC\_ALL, "rus");

  Mnvo<int> A(5, 0);

  cout <<"Проверка оператора ввода, введите ваше множество A длины 5:\n";

  cin >> A[0];

  make\_mnvo(A, 5);

  cout <<"Проверка оператора вывода, ваше множество A:\n"<< A << endl;

  Mnvo<int> B(10, 1);

  cout <<"Проверка оператора =, начальное множество B:\n"<< B << endl;

  B = A;

  cout <<"Множество B после операции B=A:\n"<< B << endl;

  cout <<"Проверка оператора [], Третий элемент множества A равен "<< A[2] << endl;

  cout <<"Проверка оператора (), Размер множества A ="<< A() << endl;

  Mnvo<int> C(3, 0);

  cout <<"Проверка оператора -, Введите множество C длины 3, которое вы хотите вычесть из множества A:\n";

  cin >> C[0];

  make\_mnvo(C, 3);

  A - C;

  cout <<"Разность множеств :\n"<< A << endl;

  // Money

  Money H;

  Mnvo<Money> X(5, H);

  cout <<"Проверка оператора ввода, введите ваше множество X длины 5:\n";

  cin >> X[0];

  make\_mnvo(X, 5);

  cout <<"Проверка оператора вывода, ваше множество X:\n"<< X << endl;

  Mnvo<Money> Y(10, H);

  cout <<"Проверка оператора =, начальное множество Y:\n"<< Y << endl;

  Y = X;

  cout <<"Множество Y после операции Y=X:\n"<< Y << endl;

  cout <<"Проверка оператора [], Третий элемент множества X равен "<< X[2] << endl;

  cout <<"Проверка оператора (), Размер множества X ="<< X() << endl;

  Mnvo<Money> Z(3, H);

  cout <<"Проверка оператора -, Введите множество Z длины 3, которое вы хотите вычесть из множества X:\n";

  cin >> Z[0];

  make\_mnvo(Z, 3);

  X - Z;

  cout <<"Разность множеств :\n"<< X << endl;

  system("pause");

}

void make\_mnvo(Mnvo<int>& A, int length)

{

  int i, j, el;

  for (i = 1; i < length; i++)

  {

    cin >> el;

    if (el > A[i - 1])

    {

      A[i] = el;

    }

    else

    {

      j = i - 1;

      while (el < A[j])

      {

        A[j + 1] = A[j];

        A[j] = 0;

        j--;

      }

      A[j + 1] = el;

    }

  }

}

void make\_mnvo(Mnvo<Money>& A, int length)

{

  int i, j;

  Money el, H;

  for (i = 1; i < length; i++)

  {

    cin >> el;

    if (el > A[i - 1])

    {

      A[i] = el;

    }

    else

    {

      j = i - 1;

      while (el < A[j])

      {

        A[j + 1] = A[j];

        A[j] = H;

        j--;

      }

      A[j + 1] = el;

    }

  }

}

**Mnvo.h**

#pragma once

#include<iostream>

usingnamespace std;

template<class T>

class Mnvo

{

public:

  Mnvo(int length, T k);

  Mnvo(Mnvo<T>&);

  ~Mnvo();

  Mnvo&operator = (Mnvo<T>&);

  intoperator () ();

  T&operator [] (int);

  Mnvo operator - (Mnvo<T>);

  friend istream&operator>><>(istream&in, Mnvo<T>&a);

  friend ostream&operator<<<>(ostream&out, const Mnvo<T>&a);

  int size;

  T\*data;

};

template<class T>

Mnvo<T>::Mnvo(int s, T k)

{

  size = s;

  data = new T[size];

  for (int i = 0; i<size; i++) data[i] = k;

}

template<class T>

Mnvo<T>::Mnvo(Mnvo& a)

{

  size = a.size;

  data = new T[size];

  for (int i = 0; i<size; i++) data[i] = a.data[i];

}

template<class T>

Mnvo<T>::~Mnvo()

{

  delete[]data;

  data = 0;

}

template<class T>

Mnvo<T>&Mnvo<T>::operator=(Mnvo<T>&a)

{

  if (this == &a) return \*this;

  size = a.size;

  if (data != 0) delete[]data;

  data = new T[size];

  for (int i = 0; i<size; i++) data[i] = a.data[i];

  return \*this;

}

template<class T>

T& Mnvo<T>::operator [] (int index)

{

  if (index<size) return data[index];

  else cout <<"\nIndex>size";

  return data[0];

}

template<class T>

int Mnvo<T>::operator ()()

{

  return size;

}

template<class T>

Mnvo<T> Mnvo<T>::operator - (Mnvo<T> a)

{

  bool f;

  int y = (\*this).size, i;

for (i = 0; i<a.size; i++)

  {

    f = false;

    for (int j = 0; j<(\*this).size; j++)

    {

      if (a[i] == (\*this)[j] && !f)

      {

        f = true;

        for (int p = j+1; p < (\*this).size; p++)

        {

         (\*this)[p-1] = (\*this)[p];

}

        size--;

      }

    }

  }

  return (\*this);

}

template<class T>

ostream&operator<< (ostream&out, const Mnvo<T>&a)

{

  for (int i = 0; i<a.size; ++i)

  {

    out << a.data[i] <<" ";

  }

  return out;

}

template<class T>

istream&operator>> (istream&in, Mnvo<T>&a)

{

  for (int i = 0; i<a.size; ++i) in>> a.data[i];

  returnin;

}

**Money.h**

#pragma once

#include<iostream>

usingnamespace std;

class Money

{

  long rub;

  int kop;

public:

  Money()

  {

    rub = 0;

    kop = 0;

  }

  Money(long r, int k)

  {

    if (k<100)

    {

      rub = r;

      kop = k;

    }

    else

    {

      rub = r + k / 100;

      kop = k % 100;

    }

    if (r <0 || k <0)

    {

      cout <<"Ошибка, количество рублей и копеек должно быть > 0";

      rub = 0;

      kop = 0;

    }

  }

  Money(Money &m)

  {

    rub = m.rub;

    kop = m.kop;

  }

  ~Money() {}

  long get\_rub()

  {

    return rub;

  }

  int get\_kop()

  {

    return kop;

  }

  void set\_rub(long r)

  {

    rub = r;

  }

  void set\_kop(int k)

  {

    kop = k;

  }

  Money&operator--();

  Money operator--(int);

  Money operator= (int&);

  booloperator==(Money&);

  booloperator==(int&);

booloperator!=(Money &);

  booloperator>(Money&);

  booloperator<(Money&);

Money operator+(Money&);

  friend istream&operator>> (istream&in, Money&m);

  friend ostream&operator<<(ostream&out, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

  kop = 0;

  return \*this;

}

Money Money::operator--(int)

{

  Money m(rub, kop);

  kop = 0;

  return m;

}

bool Money::operator==(Money &m)

{

  if (m.rub == rub && m.kop == kop)

  {

    returntrue;

  }

  else

  {

    returnfalse;

  }

}

bool Money::operator==(int&m)

{

  if (rub == m) returntrue;

  elsereturnfalse;

}

Money Money::operator= (int&m)

{

  rub = m;

  return (\*this);

}

bool Money::operator!=(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returnfalse;

}

else

{

returntrue;

}

}

bool Money::operator>(Money&m)

{

  if (rub < m.rub)

  {

    returnfalse;

  }

  if (rub > m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop > m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

}

bool Money::operator<(Money&m)

{

  if (rub > m.rub)

  {

    returnfalse;

  }

  if (rub < m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop < m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

}

Money Money::operator+(Money& m)

{

Money tmp;

tmp.rub=rub+m.rub;

if (kop+m.kop>=100)

{

tmp.kop=kop+m.kop-100;

tmp.rub++;

}

else

{

tmp.kop=kop+m.kop;

}

return tmp;

}

istream&operator>> (istream&in, Money& m)

{

  cout <<"Рубли: ";

  in>> m.rub;

  cout <<"Копейки: ";

  in>> m.kop;

  returnin;

}

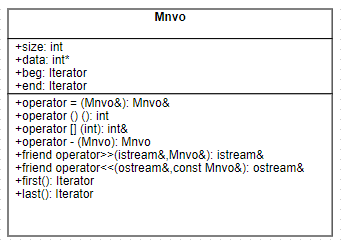
ostream&operator<<(ostream&out, const Money&m)

{

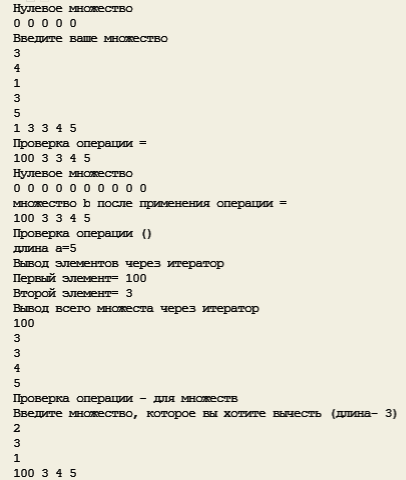
  return (out << m.rub <<","<< m.kop);

}

**Диаграмма классов**

****

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**В чем смысл использования шаблонов?**

С помощью шаблона функций можно отделить алгоритм от конкретных типов данных, передавая тип в качестве параметра.

**Каковы синтаксис/семантика шаблонов функций?**

template <class параметр> заголовокФункции

{тело функции}

**Каковы синтаксис/семантика шаблонов классов?**

template <class параметр> class имяКласса

{…};

**Что такое параметры шаблона функции?**

Типы данных, передаваемые в функцию

**Перечислите основные свойства параметров шаблона функции.**

Шаблоны методов (функций) не могут быть виртуальными.

**Как записывать параметр шаблона?**

template <class T>

**Можно ли перегружать параметризованные функции?**

Да.

**Перечислите основные свойства параметризованных классов.**

Шаблоны классов могут содержать статические элементы, дружественные функции и классы.

Шаблоны могут быть производными как от шаблонов, так и от обычных классов, а также являться базовыми и для шаблонов, и для обычных классов.

**Все ли компонентные функции параметризованного класса являются параметризованными?**

Да.

**Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?**

Нет.

**Могут ли шаблоны классов содержать виртуальные компонентные функции?**

Нет.

**Как определяются компонентные функции параметризованных классов вне определения шаблона класса?**

Такое определение невозможно.

**Что такое инстанцирование шаблона?**

Процесс генерации компилятором определения конкретного класса по шаблону класса и аргументам шаблона.

**На каком этапе происходит генерирование определения класса по шаблону?**

В момент создания экземпляра класса.

## 18.8

**Постановка задачи**

(Вариант 10)

1.  Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.

2.  Реализовать конструкторы, деструктор, операцию присваивания, селекторы и модификаторы.

3.  Определить класс-группу на основе структуры, указанной в варианте.

4.  Для группы реализовать конструкторы, деструктор, методы для добавления и удаления элементов в группу, метод для просмотра группы, перегрузить операцию для получения информации о размере группы.

5.  Определить класс Диалог – наследника группы, в котором реализовать методы для обработки событий.

6.  Добавить методы для обработки событий группой и объектами пользовательских классов.

7.  Написать тестирующую программу.

8.  Нарисовать диаграмму классов и диаграмму объектов

Базовый класс: ПЕЧАТНОЕ\_ИЗДАНИЕ(PRINT)

Название– string Автор – string Производный класс

ЖУРНАЛ (MAGAZIN)

Количество страниц - int Группа – Список (List). Команды:

• Создать группу (формат команды: m количество элементов группы).

• Добавить элемент в группу (формат команды: +)

• Удалить элемент из группы (формат команды -)

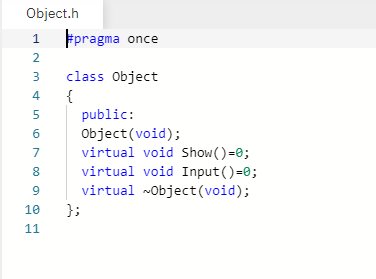
• Вывести информацию об элементах группы (формат команды: s)

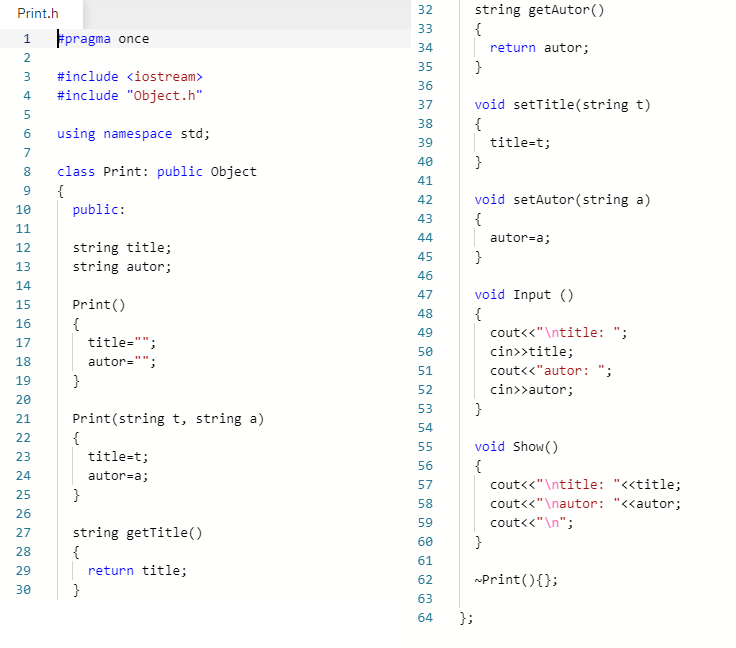
• Конец работы (формат команды: q)

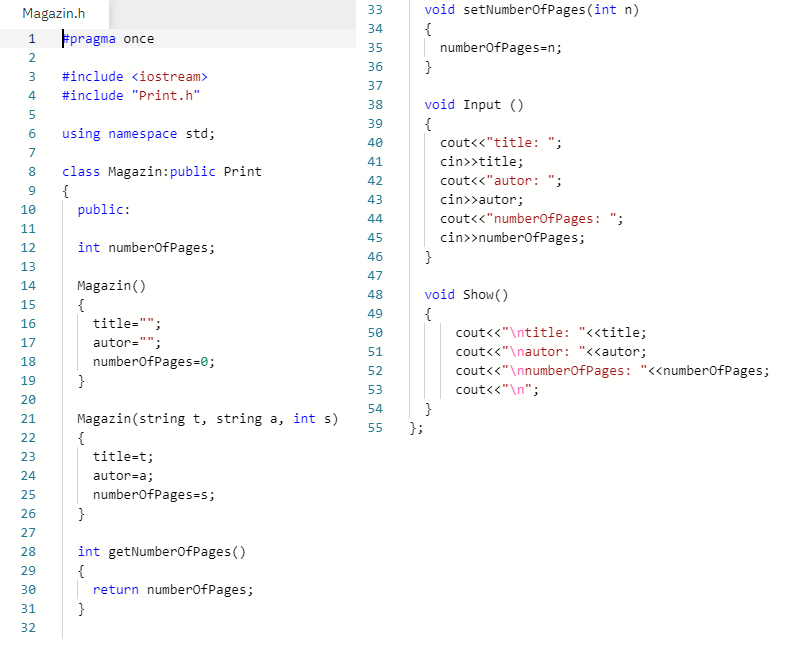
**Анализ задачи**

**1.** Для решения задачи необходимо:

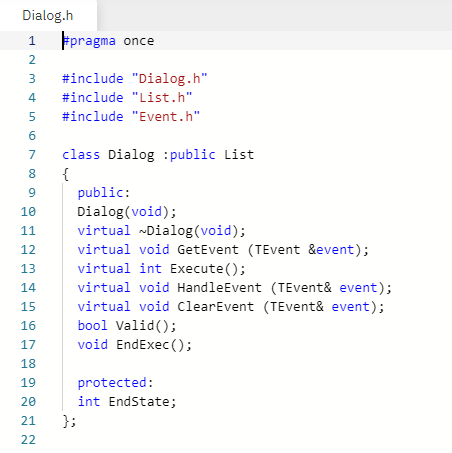
1.1.  Определить иерархию пользовательских классов (см. лабораторную работу №5). Во главе иерархии должен стоять абстрактный класс с чисто виртуальными методами для ввода и вывода информации об атрибутах объектов.







1.2.  Определить класс Диалог – наследника группы, в котором реализовать методы для обработки событий.



1.3.  Добавить методы для обработки событий группой и объектами пользовательских классов.





**2.**В ходе работы были использованы следующие типы данных:

2.1. string – для хранения фио автора и названия издания.

2.2. int – для хранения количества страниц издания.

**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Данные об изданиях были представлены в виде классов Pringи Magazin.





**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Операция ввода - cin.

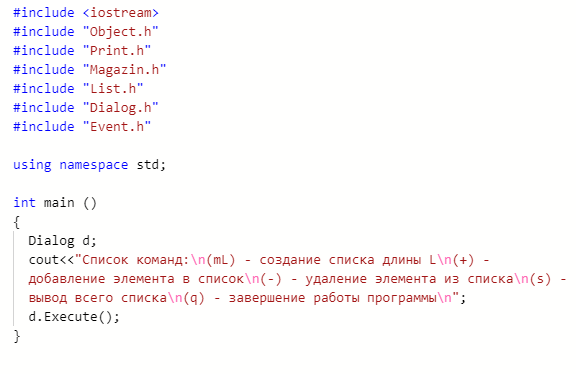


4.2. Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код для запуски основного цикла диалога.



**Решение**

**Main .cpp**

#include<iostream>

#include"Object.h"

#include"Print.h"

#include"Magazin.h"

#include"List.h"

#include"Dialog.h"

#include"Event.h"

usingnamespace std;

int main ()

{

Dialog d;

cout<<"Список команд:\n(mL) - создание списка длины L\n(+) - добавление элемента в список\n(-) - удаление элемента из списка\n(s) - вывод всего списка\n(q) - завершение работы программы\n";

d.Execute();

}

**Dialog.h**

#pragma once

#include"Dialog.h"

#include"List.h"

#include"Event.h"

class Dialog :public List

{

public:

Dialog(void);

virtual ~Dialog(void);

virtualvoid GetEvent (TEvent &event);

virtualint Execute();

virtualvoid HandleEvent (TEvent&event);

virtualvoid ClearEvent (TEvent&event);

bool Valid();

void EndExec();

protected:

int EndState;

};

**Dialog.cpp**

#include"Dialog.h"

#include"List.h"

#include"Event.h"

usingnamespace std;

Dialog::Dialog(void):List()

{

EndState=0;

}

Dialog::~Dialog(void)

{

}

void Dialog::GetEvent(TEvent &event)

{

string s;

string param;

char code;

cout<<"Вашакомада:";

cin>>s;

code = s[0];

if(code=='m' || code=='+' || code=='-' || code=='s' || code=='z' || code=='q')

{

event.what = evMessage;

switch(code)

{

case'm':

event.command=cmMake;

break;

case'+':

event.command=cmAdd;

break;

case'-':

event.command=cmDel;

break;

case's':

event.command=cmShow;

break;

case'q':

event.command = cmQuit;

break;

}

if(s.length()>1)

{

param=s.substr(1,s.length()-1);

int A=atoi(param.c\_str());

event.a=A;

}

}

else

{

event.what=evNothing;

cout<<"Невернаякоманда\n";

}

}

int Dialog::Execute()

{

TEvent event;

do

{

EndState=0;

GetEvent(event);

HandleEvent(event);

} while(!Valid());

return EndState;

}

bool Dialog::Valid()

{

if (EndState == 0) returnfalse;

elsereturntrue;

}

void Dialog::ClearEvent(TEvent&event)

{

event. what= evNothing;

}

void Dialog::EndExec()

{

EndState= 1;

}

void Dialog::HandleEvent(TEvent&event)

{

if( event.what == evMessage)

{

switch( event.command )

{

case cmMake:

if (event.a<1)

{

cout<<"Невозможно создать список указанной длины\n";

}

else

{

size=event.a;

beg=new Object\*[size];

cur=0;

cout<<"Список создан, его длина = "<<size<<'\n';

}

ClearEvent(event);

break;

case cmAdd:

Add();

ClearEvent( event );

break;

case cmDel:

Del();

ClearEvent( event );

break;

case cmShow:

if (size>0)

{

cout<<"Вашсписок:\n";

Show();

}

else

{

cout<<"Списокпуст\n";

}

ClearEvent( event );

break;

case cmQuit:

EndExec(); //выход

ClearEvent( event );

break;

};

};

}

**List.h**

#pragma once

#include"Object.h"

#include"Print.h"

#include"Magazin.h"

class List

{

public:

List(){};

List(int);

~List(void);

void Add();

void Del();

void Show();

intoperator()();

protected:

Object\*\*beg;

int size=0;

int cur;

};

**List.cpp**

#include"List.h"

usingnamespace std;

List::~List(void)

{

if(beg!=0)delete [] beg;

beg=0;

}

List::List(int n)

{

if (n>0)

{

beg=new Object\*[n]; cur=0;

size=n;

}

else

{

cout<<"Невозможно создать список указанной длины\n";

}

}

void List::Add ()

{

Object\*p;

if (size==0)

{

cout<<"Длина списка равна 0\n";

}

else

{

cout<<"1.Print"<<endl;

cout<<"2.Magazin"<<endl;

int y;

cin>>y;

if(y==1)

{

Print\*a=new (Print);

a->Input();

p=a;

if(cur<size)

{

beg[cur]=p;

cur++;

}

}

else

if(y==2)

{

Magazin \*b=new Magazin; b->Input();

p=b;

if(cur<size)

{

beg[cur]=p; cur++;

}

}

cout<<"Элементдобавлен\n";

}

}

void List::Show()

{

if(cur==0)cout<<"Длинаспискаравна 0\n";

Object \*\*p=beg;

for(int i=0;i<cur;i++)

{

(\*p)->Show();

p++;

}

}

int List::operator ()()

{

return cur;

}

void List::Del()

{

if (size>0 and cur>0)

{

cout<<"Элементудален\n";

cur--;

}

else

{

if (size<1)

{

cout<<"Длинаспискаравна 0\n";

}

if (cur<1)

{

cout<<"Списокпуст\n";

}

}

}

**Print.h**

#pragma once

#include<iostream>

#include"Object.h"

usingnamespace std;

class Print: public Object

{

public:

string title;

string autor;

Print()

{

title="";

autor="";

}

Print(string t, string a)

{

title=t;

autor=a;

}

string getTitle()

{

return title;

}

string getAutor()

{

return autor;

}

void setTitle(string t)

{

title=t;

}

void setAutor(string a)

{

autor=a;

}

void Input ()

{

cout<<"\ntitle: ";

cin>>title;

cout<<"autor: ";

cin>>autor;

}

void Show()

{

cout<<"\ntitle: "<<title;

cout<<"\nautor: "<<autor;

cout<<"\n";

}

~Print(){};

};

**Magazin.h**

#pragma once

#include<iostream>

#include"Print.h"

usingnamespace std;

class Magazin:public Print

{

public:

int numberOfPages;

Magazin()

{

title="";

autor="";

numberOfPages=0;

}

Magazin(string t, string a, int s)

{

title=t;

autor=a;

numberOfPages=s;

}

int getNumberOfPages()

{

return numberOfPages;

}

void setNumberOfPages(int n)

{

numberOfPages=n;

}

void Input ()

{

cout<<"title: ";

cin>>title;

cout<<"autor: ";

cin>>autor;

cout<<"numberOfPages: ";

cin>>numberOfPages;

}

void Show()

{

cout<<"\ntitle: "<<title;

cout<<"\nautor: "<<autor;

cout<<"\nnumberOfPages: "<<numberOfPages;

cout<<"\n";

}

};

**Event.h**

#pragma once

constint evNothing=0;

constint evMessage=100;

constint cmAdd=1;

constint cmDel=2;

constint cmGet=3;

constint cmShow=4;

constint cmMake=6;

constint cmQuit=101;

struct TEvent

{

int what;

union

{

int command;

struct

{

int message;

int a;

};

};

};

**Object.h**

#pragma once

class Object

{

public:

Object(void);

virtualvoid Show()=0;

virtualvoid Input()=0;

virtual ~Object(void);

};

**Object.cpp**

#include"Object.h"

Object::Object(void)

{

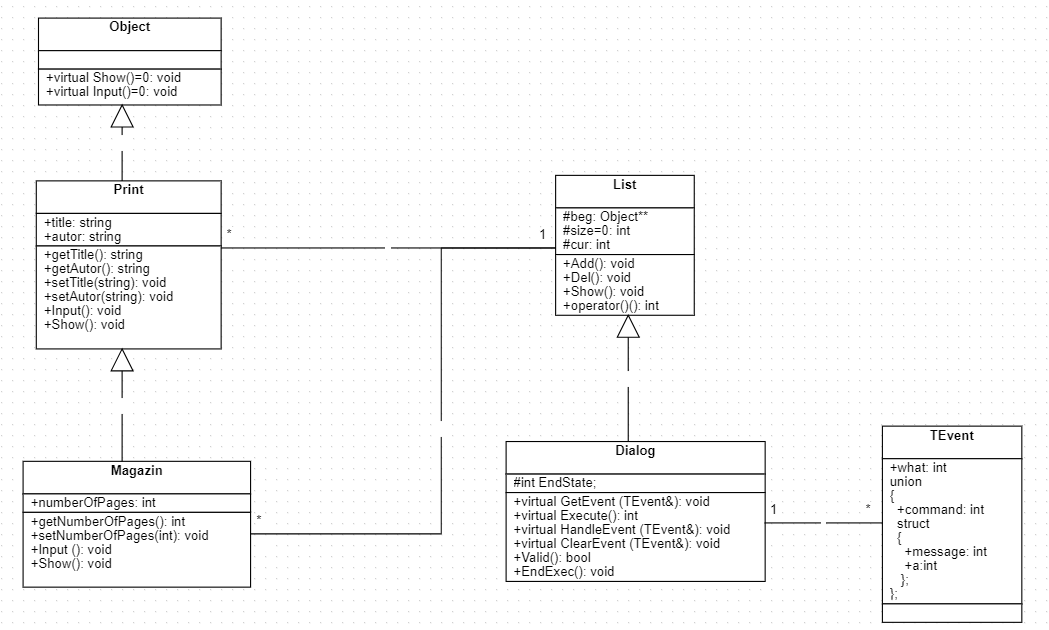
}

Object::~Object(void)

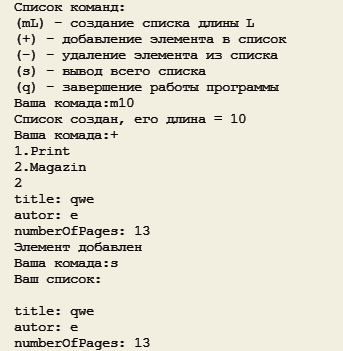
{

}

**Диаграмма классов**



**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Что такое класс-группа? Привести примеры таких классов.**

это объект, в который включены другие объекты. Объекты, входящие в группу, называются элементами группы. Элементы группы, в свою очередь, могут быть группой. Привер: ФАКУЛЬТЕТ, КАФЕДРА, СТУДЕНЧЕСКАЯ ГРУППА.

**Привести пример описания класса-группы Список (List).**

class List

{

public:

List(){};

List(int);

~List(void);

void Add();

void Del();

void Show();

int operator()();

protected:

Object\*\*beg;

int size=0;

int cur;

};

**Привести пример конструктора (с параметром, без параметров, копирования) для класса-группы Список.**

List::List(int n)

{

if (n>0)

{

beg=new Object\*[n]; cur=0;

size=n;

}

else

{

cout<<"Невозможно создать список указанной длины\n";

}

}

List::List()

{

beg=new Object\*[1];

cur=0;

size=1;

}

List::List(Object\*\* obj)

{

beg=obj;

cur=0;

size=obj->size;

}

**Привести пример деструктора для класса-группы Список.**

Object\*\*beg;

List::~List(void)

{

if(beg!=0)delete [] beg;

beg=0;

}

**Привести пример метода для просмотра элементов для класса-группы Список.**

void List::Show()

{

if(cur==0)cout<<"Длина списка равна 0\n";

Object \*\*p=beg;

for(int i=0;i<cur;i++)

{

(\*p)->Show();

p++;

}

}

**Какой вид иерархии дает группа?**

Агрегация.

**Почему во главе иерархии классов, содержащихся в группе объектов должен находиться абстрактный класс?**

Для того, чтобы была возможность обращаться к методам классов по указателю.

**Что такое событие? Для чего используются события?**

Пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя.

**Какие характеристики должно иметь событие-сообщение?**

код класса сообщения, отличающий сообщения объектов одного класса от объектов другого класса;

адрес объекта, которому предназначено сообщение (м. б. не задан, тогда сообщение могут прочитать все объекты);

информационное поле.

**Привести пример структуры, описывающей событие.**

#pragma once

const int evNothing=0;

const int evMessage=100;

const int cmAdd=1;

const int cmDel=2;

const int cmGet=3;

const int cmShow=4;

const int cmMake=6;

const int cmQuit=101;

struct TEvent

{

int what;

union

{

int command;

struct

{

int message;

int a;

};

};

};

**Задана структура события**

**struct TEvent**

**{**

**int what;**

**union**

**{**

**MouseEventType mouse;**

**KeyDownEvent keyDown;**

**MessageEvent message;**

**}**

**};**

**Какие значения, и в каких случаях присваиваются полю what?**

mouse – Движение мышью

keyDown – Нажатие кнопки вниз

message – Введено сообщение

**Задана структура события**

**struct TEvent**

**{**

**int what;//тип события union**

**{**

**int command;//код комманды**

**struct//параметры команды**

**{**

**int message;**

**int a;**

**};**

**};**

**};**

**Какие значения, и в каких случаях присваиваются полю command?**

Command – введена команда

Message – введено сообщение

**Задана структура события**

**struct TEvent**

**{**

**int what;//тип события union**

**{**

**int command;//код комманды struct//параметры команды**

**{**

**int message; int a;**

**};**

**};**

**};**

**Для чего используются поля a и message?**

Для передачи дополнительной информации

**Какие методы необходимы для организации обработки сообщений?**

GetEvent – формирование события;

Execute реализует главный цикл обработки событий. Он постоянно получает событие путем вызова GetEvent и обрабатывает их с помощью HandleEvent. Этот цикл завершается, когда поступит событие «конец».

HandleEvent – обработчик событий. Обрабатывает каждое событие нужным для него образом. Если объект должен обрабатывать определенное событие (сообщение), то его метод HandleEvent должен распознавать это событие и реагировать на него должным образом. Событие может распознаваться, например, по коду команды (поле command).

ClearEvent очищает событие, когда оно обработано, чтобы оно не обрабатывалось далее.

Обработчик событий (метод HandleEvent).

**Какой вид имеет главный цикл обработки событий-сообщений?**

Главный цикл обработки событий реализуется в методе Execute главной группы- объекта “прикладная программа” по следующей схеме:

int TMyApp::Execute()

{

do

{

endState=0;

GetEvent(event);

HandleEvent(event);

if(event.what!=evNothing) EventError(event);

}

while(!Valid());

return endState;

}

**Какую функцию выполняет метод ClearEvent()? Каким образом?**

ClearEvent очищает событие, присваивая полю event.What значение evNothing.

**Какую функцию выполняет метод HandleEvent ()? Каким образом?**

Метод HandleEvent программы обрабатывает событие “конец работы”, вызывая метод EndExec. EndExec изменяет значение private – переменной EndState. Значение этой переменной проверяет метод–функция Valid, возвращающая значение true, если “конец работы”.

**Какую функцию выполняет метод GetEvent ()?**

Получает событие и передает его в Execute()

**Для чего используется поле EndState?**

Значение переменной EndState отвечает за окончание выполнения основного цикла.

**Какой класс (объект) содержит это поле?**

Класс Dialog (Основной класс обработки событий)

**Для чего используется функция Valid()?**

Проверяет значение EndState, возвращающает значение true, если “конец работы”.

## 18.9

**Постановка задачи**

(Вариант 10)

1.  Реализовать класс, перегрузить для него операции, указанные в варианте.

2.  Определить исключительные ситуации.

3.  Предусмотреть генерацию исключительных ситуаций.

Класс- контейнер МНОЖЕСТВО с элементами типа int. Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора;

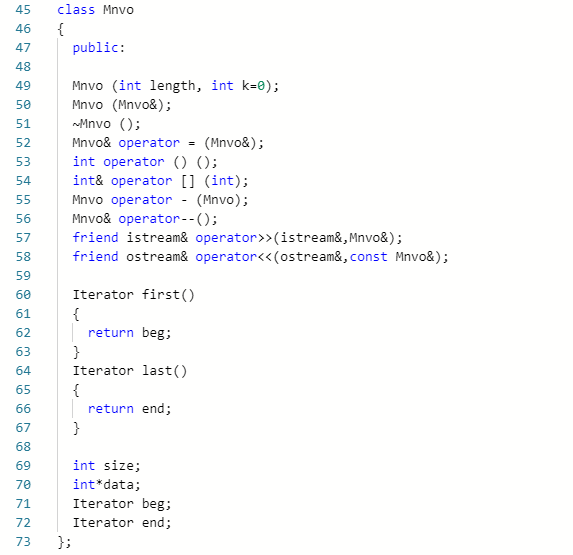
- – разность множеств;

-- – удаление элемента из множества.

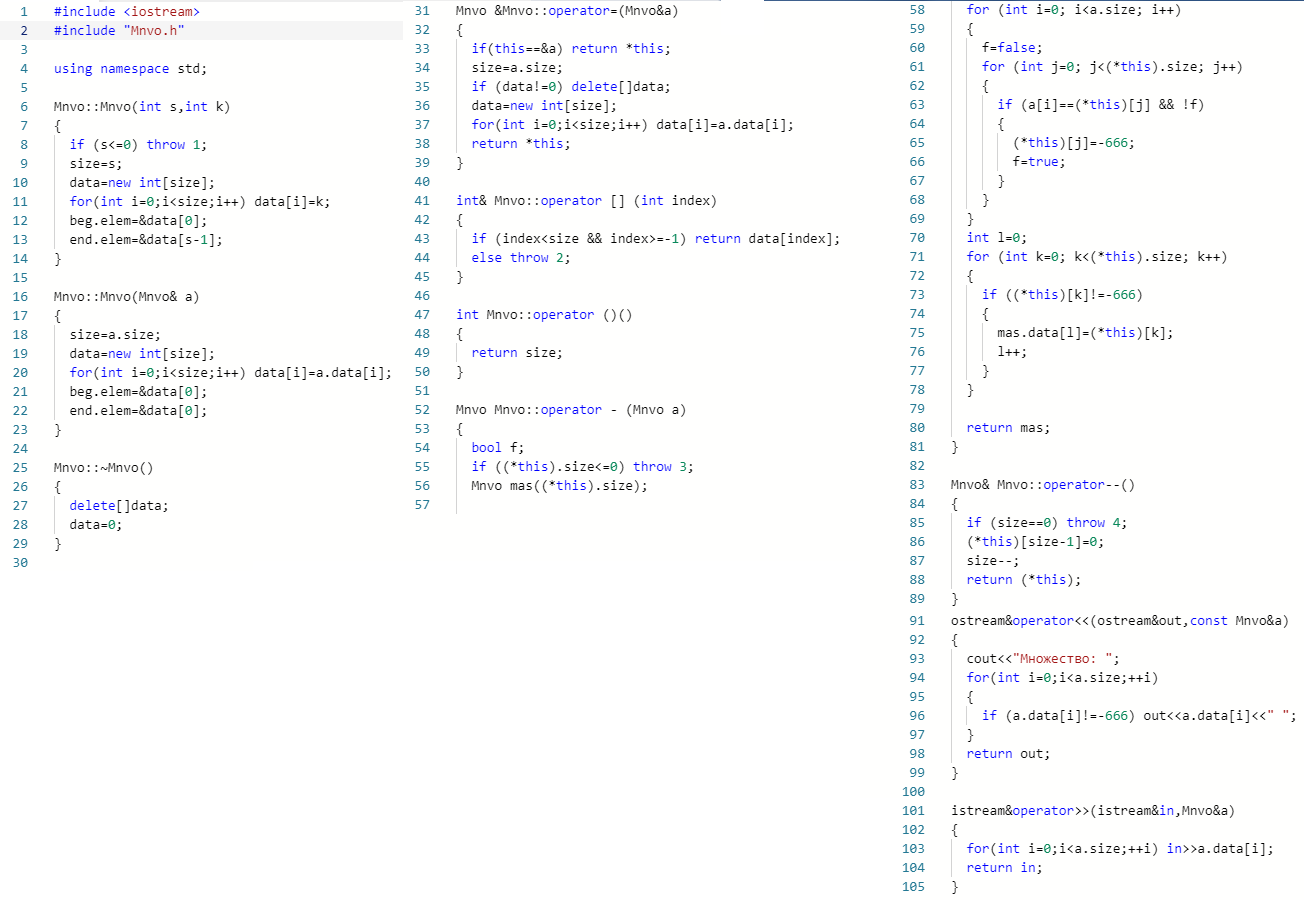
**Анализ задачи**

1. Для решения задачи необходимо:

1.1.  Реализовать класс, перегрузить для него операции, указанные в варианте.



1.2.  Предусмотреть генерацию исключительных ситуаций.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Тип int для хранения значений элеменов.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Данные об элементах хранились внутри класса Mnvo.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Операция ввода - cin.



4.2. Операция вывода - cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код для демонстрации работы обработки исключительных ситуаций.

**Решение**

**Main.cpp**

#include<iostream>

#include"Mnvo.h"

usingnamespace std;

void make\_mnvo(Mnvo&, int);

int main ()

{

try

{

int size, ind,move;

cout<<"Введите размер вашего множества: ";

cin>>size;

Mnvo mas(size);

cout<<"Введите ваше множество\n";

cin>>mas[0];

make\_mnvo(mas,size);

cout<<mas;

cout<<"\n1- Вывод элемента по индексу\n2- Разность множеств\n3- Уменьшение массива\nВаша команда: ";

cin>>move;

switch (move)

{

case1:

cout<<"\nВведите индекс элемента, который вы хотите вывести: ";

cin>>ind;

cout<<mas[ind]<<" - вашэлемент\n";

break;

case2:

cout<<"Введите размер вашего множества, которое вы хотите вычесть: ";

cin>>size;

Mnvo newM(size);

cout<<"Введите ваше множество, которое вы хотите вычесть: ";

cin>>newM[0];

make\_mnvo(newM,size);

cout<<mas<<"\n-\n"<<newM<<"\n=\n";

mas-newM;

cout<<mas<<'\n';

break;

}

if (move ==3)

{

cout<<"Уменьшение массива до появления ошибки:\n";

while (true)

{

cout<<mas<<'\n';

--mas;

}

}

if (move!=1&&move!=2&& move!=3) throw5;

}

catch (int)

{

cout<<"Ошибка";

}

}

void make\_mnvo(Mnvo& A, int length)

{

int i,j,el;

for (i = 1; i < length; i++)

  {

    cin >> el;

    if (el > A[i-1])

    {

      A[i] = el;

    }

    else

    {

      j = i-1;

      while (el < A[j])

      {

        A[j + 1] = A[j];

        A[j] = 0;

        j--;

     }

      A[j + 1] = el;

    }

  }

}

**Mnvo.h**

#include<iostream>

usingnamespace std;

class Iterator

{

friendclass Mnvo;

public:

Iterator()

{

elem=0;

}

Iterator(const Iterator&it)

{

elem=it.elem;

}

booloperator==(const Iterator&it)

{

return elem==it.elem;

}

booloperator!=(const Iterator&it)

{

return elem!=it.elem;

};

voidoperator++()

{

++elem;

};

voidoperator--()

{

--elem;

}

int&operator \*() const

{

return\*elem;

}

int \*elem;

};

class Mnvo

{

public:

Mnvo (int length, int k=0);

Mnvo (Mnvo&);

~Mnvo ();

Mnvo&operator = (Mnvo&);

intoperator () ();

int&operator [] (int);

Mnvo operator - (Mnvo);

Mnvo&operator--();

friend istream&operator>>(istream&,Mnvo&);

friend ostream&operator<<(ostream&,const Mnvo&);

Iterator first()

{

return beg;

}

Iterator last()

{

return end;

}

int size;

int\*data;

Iterator beg;

Iterator end;

};

**Mnvo.cpp**

#include<iostream>

#include"Mnvo.h"

usingnamespace std;

Mnvo::Mnvo(int s,int k)

{

if (s<=0) throw1;

size=s;

data=newint[size];

for(int i=0;i<size;i++) data[i]=k;

beg.elem=&data[0];

end.elem=&data[s-1];

}

Mnvo::Mnvo(Mnvo& a)

{

size=a.size;

data=newint[size];

for(int i=0;i<size;i++) data[i]=a.data[i];

beg.elem=&data[0];

end.elem=&data[0];

}

Mnvo::~Mnvo()

{

delete[]data;

data=0;

}

Mnvo &Mnvo::operator=(Mnvo&a)

{

if(this==&a) return \*this;

size=a.size;

if (data!=0) delete[]data;

data=newint[size];

for(int i=0;i<size;i++) data[i]=a.data[i];

return \*this;

}

int& Mnvo::operator [] (int index)

{

if (index<size && index>=-1) return data[index];

elsethrow2;

}

int Mnvo::operator ()()

{

return size;

}

Mnvo Mnvo::operator - (Mnvo a)

{

bool f;

if ((\*this).size<=0) throw3;

Mnvo mas((\*this).size);

for (int i=0; i<a.size; i++)

{

f=false;

for (int j=0; j<(\*this).size; j++)

{

if (a[i]==(\*this)[j] && !f)

{

(\*this)[j]=-666;

f=true;

}

}

}

int l=0;

for (int k=0; k<(\*this).size; k++)

{

if ((\*this)[k]!=-666)

{

mas.data[l]=(\*this)[k];

l++;

}

}

return mas;

}

Mnvo& Mnvo::operator--()

{

if (size==0) throw4;

(\*this)[size-1]=0;

size--;

return (\*this);

}

ostream&operator<<(ostream&out,const Mnvo&a)

{

cout<<"Множество: ";

for(int i=0;i<a.size;++i)

{

if (a.data[i]!=-666) out<<a.data[i]<<" ";

}

return out;

}

istream&operator>>(istream&in,Mnvo&a)

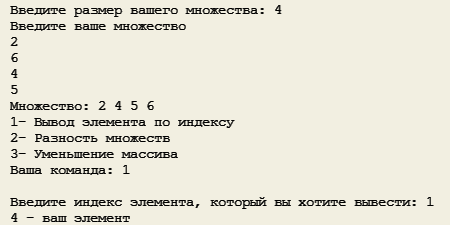
{

for(int i=0;i<a.size;++i) in>>a.data[i];

returnin;

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Что представляет собой исключение в С++?**

Исключение – это непредвиденное или аварийное событие.

**На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?**

Исключения позволяют разделить вычислительный процесс на 2 части:

обнаружение аварийной ситуации (неизвестно как обрабатывать);

обработка аварийной ситуации (неизвестно, где она возникла).

Достоинства такого подхода:

удобно использовать в программе, которая состоит из нескольких модулей;

не требуется возвращать значение в вызывающую функцию

**Какой оператор используется для генерации исключительной ситуации?**

throw <выражение>, где <выражение> -

либо константа,

либо переменная некоторого типа,

либо выражение некоторого типа.

**Что представляет собой контролируемый блок? Для чего он нужен?**

try

{

….

//ошибка

throw выражение

//генерируется исключение

Его задача: обнаружение аварийной ситуации.

**Что представляет собой секция-ловушка? Для чего она нужна?**

catch (……)

Её задача: обработка аварийной ситуации

**Какие формы может иметь спецификация исключения в секции ловушке? В каких ситуациях используются эти формы?**

Спецификация исключения может иметь три формы:

(тип имя)

(тип)

(…)

Тип – это встроенный тип или тип, определенный программистом.

Формы 1 и 2 обрабатывают конкретные исключения, а форма 3 перехватывает все исключения, такую ловушку надо помещать последней, тогда она будет обрабатывать все исключения, которые еще не были обработаны.

**Какой стандартный класс можно использовать для создания собственной иерархии исключений?**

Класс exception;

**Каким образом можно создать собственную иерархию исключений?**

Для создания собственной иерархии исключений надо объявить свой базовый класс- исключение, например:

class BaseException{};

Остальные классы будут наследниками этого класса, аналогично тому, как это сделано в иерархии стандартных исключений:

class Child\_Exception1:public BaseException{};

class Child\_Exception2:public BaseException{};

Класс BaseException можно унаследовать от стандартного класса exception class BaseException: public exception{};

**Если спецификация исключений имеет вид:**

**void f1()throw(int,double); то какие исключения может прождать функция f1()?**

Исключения типа int и double.

**Если спецификация исключений имеет вид: void f1()throw(); то какие исключения может прождать функция f1()?**

Исключения любых типов.

**В какой части программы может генерироваться исключение?**

Внутри блока try{}

**Написать функцию, которая вычисляет площадь треугольника по трем сторонам (формула Герона).**

**Функцию реализовать в 4 вариантах:**

**без спецификации исключений;**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

int length1,length2,length3;

double p,s;

cout<<"Введите длины трех сторон треугольниа, для которого вы хотите найти площадь\n";

cin>>length1>>length2>>length3;

p=(length1+length2+length3)/2;

s=sqrt(p\*(p-length1)\*(p-length2)\*(p-length3));

cout<<"S= "<<s;

}

**со спецификацией throw();**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

int length1,length2,length3;

double p,s;

cout<<"Введите длины трех сторон треугольниа, для которого вы хотите найти площадь\n";

cin>>length1>>length2>>length3;

try

{

if (length1>length2+length3 || length2>length1+length3 || length3>length2+length1 ) throw (1);

p=(length1+length2+length3)/2;

s=sqrt(p\*(p-length1)\*(p-length2)\*(p-length3));

cout<<"S= "<<s;

}

catch(int)

{

cout<<"Ошибка";

}

}

**с конкретной спецификацией с подходящим стандартным исключением**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

int length1,length2,length3;

double p,s;

cout<<"Введите длины трех сторон треугольниа, для которого вы хотите найти площадь\n";

cin>>length1>>length2>>length3;

try

{

p=(length1+length2>+length3)/2;

s=sqrt(p\*(p-length1)\*(p-length2)\*(p-length3));

cout<<"S= "<<s;

}

catch(exception& e)

{

e.what();

}

}

**спецификация с собственным реализованным исключением**

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

int length1,length2,length3;

double p,s;

cout<<"Введите длины трех сторон треугольниа, для которого вы хотите найти площадь\n";

cin>>length1>>length2>>length3;

try

{

if (length1<=0 || length2<=0 || length3<=0) throw (2);

if (length1>length2+length3 || length2>length1+length3 || length3>length2+length1) throw (1);

s=sqrt(p\*(p-length1)\*(p-length2)\*(p-length3));

cout<<"S= "<<s;

}

catch(int n)

{

if (n==1) cout<<"Ошибка, длина одной стороны больше суммы длин двух других";

if (n==2) cout<<"Ошибка, длина не может быть отрицательной";

}

}

## 18.10

**Постановка задачи**

(Вариант 10)

1.  Создать пользовательский класс с минимальной функциональностью.

2.  Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле).

3.  Написать функцию для чтения и просмотра объектов из потока.

4.  Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

5.  Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

6.  Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

7.  Для вызова функций в основной программе предусмотреть меню.

Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой.

Реализовать:

− операции сравнения (==, !=).

− вычитание копеек (--) (постфиксная и префиксная формы)

Задание:

• Удалить все записи большие заданного значения.

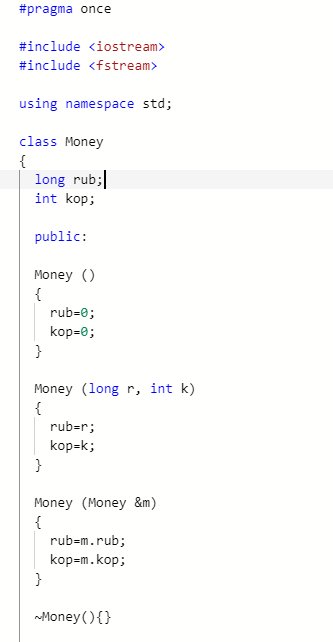
• Добавить K записей после записи с номером N.

• Увеличить все записи с заданным значением на 1 рубль 50 копеек.

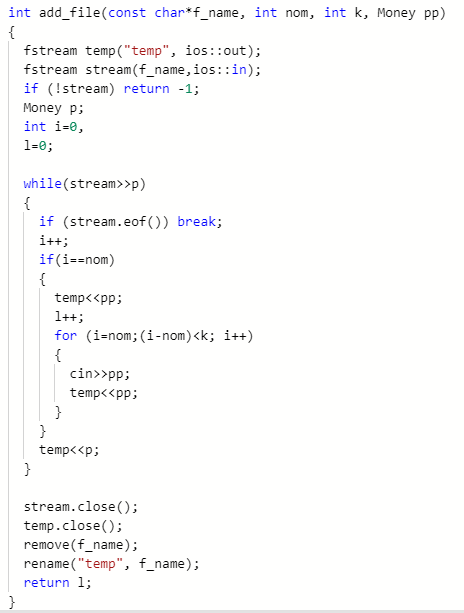
**Анализ задачи**

**1.** Для решения задачи необходимо:

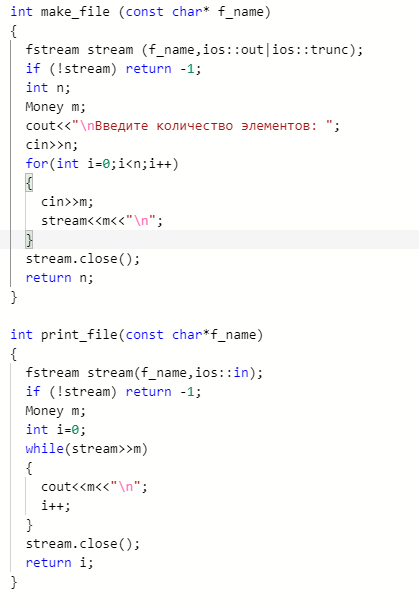
1.1.  Создать пользовательский класс с минимальной функциональностью.



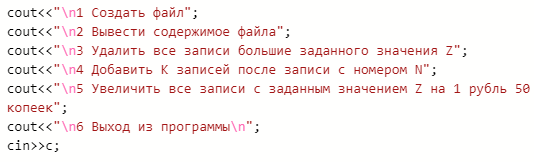
1.2.  Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле). Написать функцию для чтения и просмотра объектов из потока. Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.



1.3.  Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.



1.4.  Для вызова функций в основной программе предусмотреть меню.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения данных об элементах использовался пользовательский класс Money.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Информация об элементах хранилась в пользовательском классе Money.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.

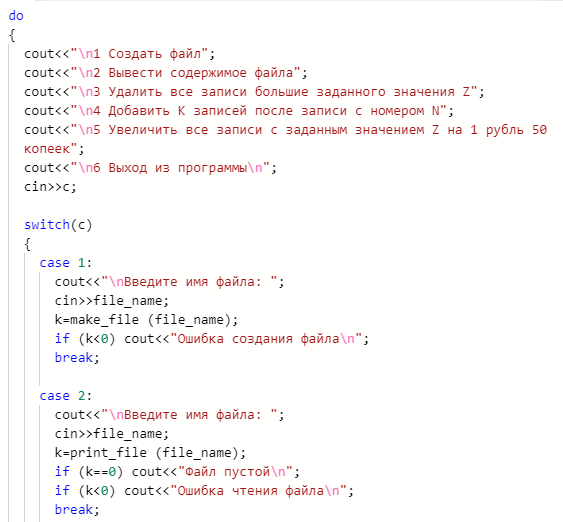


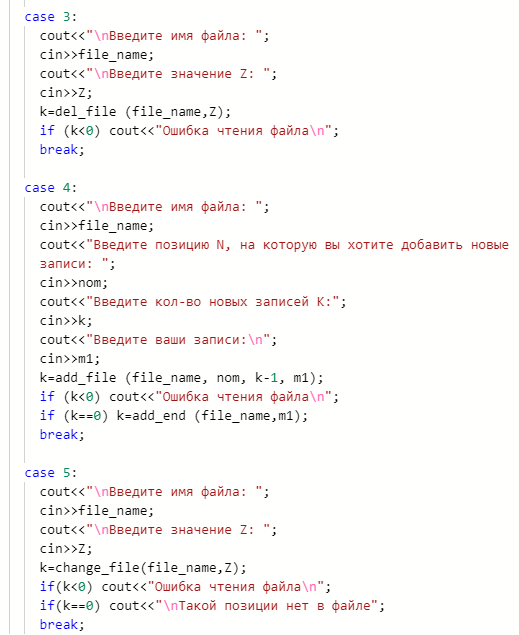
4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализовано меню для вызова методов работы с файлом







**Решение**

**Main.cpp**

#include<iostream>

#include"WorkWFile.h"

#include"Money.h"

usingnamespace std;

int main()

{

Money p, m1, m2;

Money Z;

int k,c, nom;

char file\_name[30];

do

{

cout<<"\n1 Создатьфайл";

cout<<"\n2 Вывести содержимое файла";

cout<<"\n3 Удалить все записи большие заданного значения Z";

cout<<"\n4 Добавить K записей после записи с номером N";

cout<<"\n5 Увеличить все записи с заданным значением Z на 1 рубль 50 копеек";

cout<<"\n6 Выход из программы\n";

cin>>c;

switch(c)

{

case1:

cout<<"\nВведитеимяфайла: ";

cin>>file\_name;

k=make\_file (file\_name);

if (k<0) cout<<"Ошибка создания файла\n";

break;

case2:

cout<<"\nВведитеимяфайла: ";

cin>>file\_name;

k=print\_file (file\_name);

if (k==0) cout<<"Файлпустой\n";

if (k<0) cout<<"Ошибка чтения файла\n";

break;

case3:

cout<<"\nВведитеимяфайла: ";

cin>>file\_name;

cout<<"\nВведитезначениеZ: ";

cin>>Z;

k=del\_file (file\_name,Z);

if (k<0) cout<<"Ошибка чтения файла\n";

break;

case4:

cout<<"\nВведитеимяфайла: ";

cin>>file\_name;

cout<<"ВведитепозициюN, на которую вы хотите добавить новые записи: ";

cin>>nom;

cout<<"Введите кол-во новых записей K:";

cin>>k;

cout<<"Введите ваши записи:\n";

cin>>m1;

k=add\_file (file\_name, nom, k-1, m1);

if (k<0) cout<<"Ошибка чтения файла\n";

if (k==0) k=add\_end (file\_name,m1);

break;

case5:

cout<<"\nВведитеимяфайла: ";

cin>>file\_name;

cout<<"\nВведитезначениеZ: ";

cin>>Z;

k=change\_file(file\_name,Z);

if(k<0) cout<<"Ошибка чтения файла\n";

if(k==0) cout<<"\nТакой позиции нет в файле";

break;

}

} while(c!=6);

}

**Money.h**

#pragma once

#include<iostream>

#include<fstream>

usingnamespace std;

class Money

{

long rub;

int kop;

public:

Money ()

{

rub=0;

kop=0;

}

Money (long r, int k)

{

rub=r;

kop=k;

}

Money (Money &m)

{

rub=m.rub;

kop=m.kop;

}

~Money(){}

long get\_rub ()

{

return rub;

}

int get\_kop ()

{

return kop;

}

void set\_rub (long r)

{

rub=r;

}

void set\_kop (int k)

{

kop=k;

}

Money&operator--();

Money operator--(int);

booloperator==(Money&);

booloperator!=(Money &);

booloperator<(Money&);

Money operator+(Money&);

friend istream&operator>>(istream&in, Money&m);

friend ostream&operator<<(ostream&out, const Money&m);

friend fstream&operator>>(fstream &fin, Money &m);

friend fstream&operator<<(fstream &fout, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

kop=0;

return \*this;

}

Money Money::operator--(int)

{

Money m (rub, kop);

kop=0;

return m;

}

bool Money::operator==(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returntrue;

}

else

{

returnfalse;

}

}

bool Money::operator!=(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returnfalse;

}

else

{

returntrue;

}

}

bool Money::operator< (Money&m)

{

if (rub<m.rub)

{

returntrue;

}

else

{

if (rub>m.rub)

{

returnfalse;

}

else

{

if (kop<m.kop)

{

returntrue;

}

else

{

returnfalse;

}

}

}

}

Money Money::operator+(Money& m)

{

Money tmp;

tmp.rub=rub+m.rub;

if (kop+m.kop>=100)

{

tmp.kop=kop+m.kop-100;

tmp.rub++;

}

else

{

tmp.kop=kop+m.kop;

}

return tmp;

}

istream&operator>>(istream&in, Money& m)

{

cout<<"Рубли: ";

in>>m.rub;

cout<<"Копейки: ";

in>>m.kop;

returnin;

}

ostream&operator<<(ostream&out, const Money&m)

{

return (out<<m.rub<<","<<m.kop);

}

fstream&operator>>(fstream& fin, Money&m)

{

fin>>m.rub;

fin>>m.kop;

return fin;

}

fstream&operator<<(fstream& fout, const Money &m)

{

fout<<m.rub<<"\n"<<m.kop<<"\n";

return fout;

}

**WorkWFile.h**

#pragma once

#include"Money.h"

#include<iostream>

#include<fstream>

usingnamespace std;

int make\_file (constchar\* f\_name)

{

fstream stream (f\_name,ios::out|ios::trunc);

if (!stream) return -1;

int n;

Money m;

cout<<"\nВведите количество элементов: ";

cin>>n;

for(int i=0;i<n;i++)

{

cin>>m;

stream<<m<<"\n";

}

stream.close();

return n;

}

int print\_file(constchar\*f\_name)

{

fstream stream(f\_name,ios::in);

if (!stream) return -1;

Money m;

int i=0;

while(stream>>m)

{

cout<<m<<"\n";

i++;

}

stream.close();

return i;

}

int del\_file(constchar\*f\_name, Money z)

{

fstream temp ("temp", ios::out);

fstream stream (f\_name,ios::in);

int i=0;

Money p;

while(stream>>p)

{

if (stream.eof()) break;

i++;

if (p<z || p==z) temp<<p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

int add\_file(constchar\*f\_name, int nom, int k, Money pp)

{

fstream temp("temp", ios::out);

fstream stream(f\_name,ios::in);

if (!stream) return -1;

Money p;

int i=0,

l=0;

while(stream>>p)

{

if (stream.eof()) break;

i++;

if(i==nom)

{

temp<<pp;

l++;

for (i=nom;(i-nom)<k; i++)

{

cin>>pp;

temp<<pp;

}

}

temp<<p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;

}

int add\_end(constchar\*f\_name, Money pp)

{

fstream stream(f\_name, ios::app);

if (!stream) return -1;

stream<<pp;

return1;

}

int change\_file(constchar\*f\_name, Money Z)

{

fstream temp("temp", ios::out);

fstream stream(f\_name,ios::in);

if (!stream) return -1;

Money p;

Money c(1,50);

int i=0, l=0;

while(stream>>p)

{

if (stream.eof()) break;

if(Z==p)

{

p=p+c;

temp<<p;

l++;

}

else temp<<p;

}

stream.close();

temp.close();

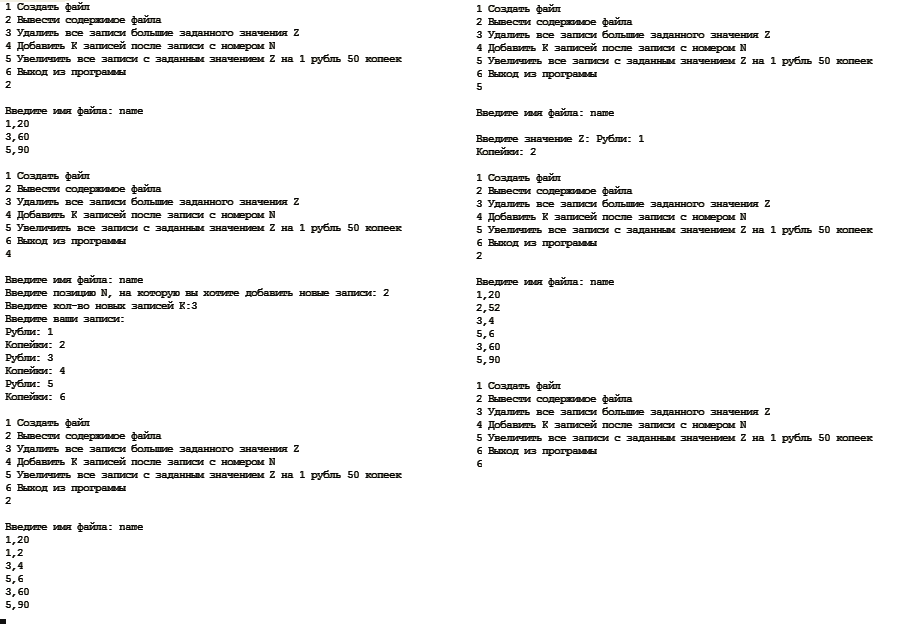
remove(f\_name);

rename("temp", f\_name);

return l;

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Что такое поток?**

Поток - определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен.

**Какие типы потоков существуют?**

Потоки бывают

Стандартные: только однонаправленные, либо входные, либо выходные.

Строковые: могут быть и однонаправленными и двунаправленными

Файловые: могут быть и однонаправленными и двунаправленными.

**Какую библиотеку надо подключить при использовании стандартных потоков?**

#include <iostream>

**Какую библиотеку надо подключить при использовании файловых потоков?**

#include <fstream>

**Какую библиотеку надо подключить при использовании строковых потоков?**

#include <sstream>

**Какая операция используется при выводе в форматированный поток?**

operator<<

**Какая операция используется при вводе из форматированных потоков?**

operator>>

**Какие методы используются при выводе в форматированный поток?**

|  |  |  |
| --- | --- | --- |
| ostream& put(char c) | char c=’a’;  stream.put(c); | Записывает в поток  stream символ c |
| ostream& write(const char\* buf, int size) | char c=’a’; stream.write(&c); | Записывает в поток stream символ c |
| ostream& write(const char\* buf, int size) | char s[]=”string1”; stream.write(s,strlen  (s)); | Записывает в поток stream строку символов |

**Какие методы используется при вводе из форматированного потока?**

|  |  |  |
| --- | --- | --- |
| Прототип | Пример |  |
| int\_type get();  istream& get(char &c) | c= stream.get();  stream.get(c) | ввод из потока одного  символа |
| istream&read(char\*buf,int  size) | stream.read(&c,1) | ввод из потока одного  символа |

Ввод символьных массивов и строк выполняется с помощью операции >> до первого символа-разделителя (обычно пробела). Для ввод строк с пробелами используют методы get() и getline().

|  |  |  |
| --- | --- | --- |
| Прототип | Пример |  |
| istream& get(char\*str,  streamsize count) | stream.get(s,50) | ввод из потока строки с  пробелами (пока не встретиться признак конца файла eof или не будет введено 50 символов) |
| istream& get(char\*str,  streamsize count, char lim) | stream.get(s,50,  ’;’) | ввод из потока строки с  пробелами (пока не встретиться признак конца файла eof, или не будет введено 50 символов, или не встретится ; ) |
| istream&  getline(char\*str, streamsize count) | stream.getline(s,  50) | Работает также как get,  но удаляет из входного потока символ ‘\n’ |
| istream& get(char\*str,  streamsize count, char lim) | stream.getline(s,  50,’;’) |

Ввод в двоичные файлы производится методом read:

istream&read(char\*buf, streamsize size)

Метод читает size символов в массив buf. Символы разделители на ввод не влияют.

**Какие режимы для открытия файловых потоков существуют?**

Режимы открытия потока

|  |  |
| --- | --- |
| Режим | Описание |
| in | открыть поток для чтения (по умолчанию для ifstream) |
| out | (по умолчанию для оfstream) |
| trunk | удалить старое содержимое файла (по умолчанию для оfstream) |
| app | открыть поток для записи в конец файла |
| ate | открыть поток для чтения и/или записи и встать в конец файла |
| binary | открыть поток в двоичном режиме |

**Какой режим используется для добавления записей в файл?**

in.

**Какой режим (комбинация режимов) используется в конструкторе ifstream** **file(“f.txt”)?**

in.

**Какой режим (комбинация режимов) используется в конструкторе fstream file(“f.txt”)?**

In|out

**Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?**

Out.

**Каким образом открывается поток в режиме ios::out|ios::app?**

fstream stream (”d:/files/number.txt”, ios::out|ios::app);

**Каким образом открывается поток в режиме ios::out |ios::trunc?**

fstream stream (”d:/files/number.txt”, ios::out|ios::trunc);

**Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?**

fstream stream (”d:/files/number.txt”, ios::out | ios::in| ios::trunk);

**Каким образом можно открыть файл для чтения?**

ifstream stream (”d:/files/number.txt”);

**Каким образом можно открыть файл для записи?**

ofstream stream (”d:/files/number.txt”);

**Привести примеры открытия файловых потоков в различных режимах.**

ofstream stream (”d:/files/number.txt”);

ifstream stream;

stream.open(”c:/files/number.txt”);

ofstream stream (”d:/files/number.txt”, std::ios::app);

**Привести примеры чтения объектов из потока.**

int nl=0;

while(stream.get(c))

{

if (c==’\n’) nl++;

}

**Привести примеры записи объектов в поток.**

stream<<”smth”;

**Сформулировать алгоритм удаления записей из файла.**

int del\_file(string f\_name, Money z)

{

ofstream temp ("temp");

ifstream stream (f\_name);

int i=0;

Money p;

while(stream>>p)

{

if (stream.eof()) break;

i++;

if (p<z || p==z) temp<<p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

**Сформулировать алгоритм добавления записей в файл.**

int add\_file(string f\_name, int nom, int k, Money pp)

{

ofstream temp("temp");

ifstream stream(f\_name);

if (!stream) return -1;

Money p;

int i=0,

l=0;

while(stream>>p)

{

if (stream.eof()) break;

i++;

if(i==nom)

{

temp<<pp;

l++;

for (i=nom;(i-nom)<k; i++)

{

cin>>pp;

temp<<pp;

}

}

temp<<p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;

}

**Сформулировать алгоритм изменения записей в файле.**

int change\_file(string \*f\_name, Money Z)

{

ofstream temp("temp");

ifstream stream(f\_name);

if (!stream) return -1;

Money p;

Money c(1,50);

int i=0, l=0;

while(stream>>p)

{

if (stream.eof()) break;

if(Z==p)

{

p=p+c;

temp<<p;

l++;

}

else temp<<p;

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;

}

## 18.11.1

**Постановка задачи**

(Вариант 10)

Задача 1.

1.  Создать последовательный контейнер.

2.  Заполнить его элементами стандартного типа (тип указан в варианте).

3.  Добавить элементы в соответствии с заданием

4.  Удалить элементы в соответствии с заданием.

5.  Выполнить задание варианта для полученного контейнера.

6.  Выполнение всех заданий оформить в виде глобальных функций.

Задача 1

1.  Контейнер - вектор

2.  Тип элементов - float

Задание 3

Найти минимальный элемент и добавить его на заданную позицию

контейнера

Задание 4

Найти элементы большие среднего арифметического и

удалить их из контейнера

Задание 5

Каждый элемент домножить на максимальный элемент контейнера

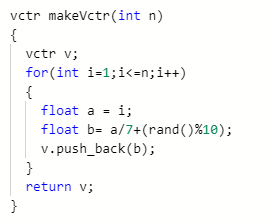
**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1.  Создать последовательный контейнер.

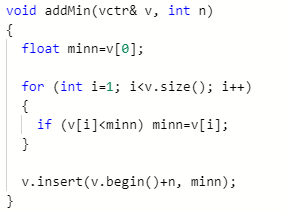


1.2.  Заполнить его элементами типаfloat.



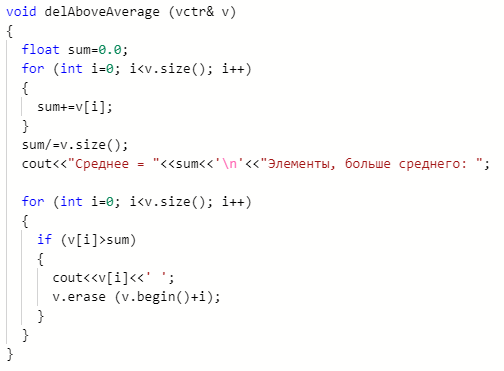
1.3.Найти минимальный элемент и добавить его на заданную позицию

контейнера .

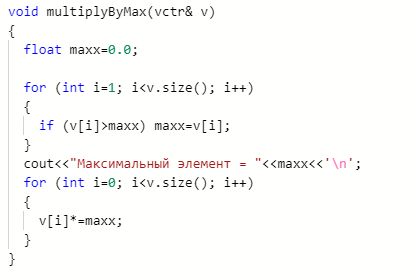


1.4. Найти элементы большие среднего арифметического и

удалить их из контейнера.



1.5. Каждый элемент домножить на максимальный элемент контейнера.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался тип float.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Массив элементов был представлен в виде вектора, с типом элементов float.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий действия над элементами вектора, используя ранее реализованные функции.



**Решение**

**Main.cpp**

#include<iostream>

#include<vector>

usingnamespace std;

typedef vector<float> vctr;

vctr makeVctr(int);

void printVctr(vctr);

void addMin(vctr&, int);

void delAboveAverage (vctr&);

void multiplyByMax(vctr&);

int main()

{

try

{

vctr v;

vctr::iterator vi=v.begin();

int n;

// 1, 2

cout<<"Введите длину вектора: ";

cin>>n;

if (n<1) throw0;

v=makeVctr(n);

printVctr(v);

//

cout<<"----------------------------------------------------------------------------\n";

// 3

cout<<"Добавление минимального элемента на позицию N, введите N:";

cin>>n;

if (n<1) throw0;

addMin(v, n);

printVctr(v);

//

cout<<"----------------------------------------------------------------------------\n";

//4

cout<<"Найти элементы большие среднего арифметического и удалить их из контейнера\n";

delAboveAverage(v);

printVctr(v);

//

cout<<"----------------------------------------------------------------------------\n";

// 5

cout<<"Каждый элемент домножить на максимальный элемент контейнера\n";

multiplyByMax(v);

printVctr(v);

//

}

catch(int)

{

cout<<"Ошибка";

}

}

vctr makeVctr(int n)

{

vctr v;

for(int i=1;i<=n;i++)

{

float a = i;

float b= a/7+(rand()%10);

v.push\_back(b);

}

return v;

}

void printVctr(vctr v)

{

cout<<"\nВектор: ";

for (int i=0; i<v.size(); i++)

{

cout<<v[i]<<" ";

}

cout<<'\n';

cout<<endl;

}

void addMin(vctr& v, int n)

{

float minn=v[0];

for (int i=1; i<v.size(); i++)

{

if (v[i]<minn) minn=v[i];

}

v.insert(v.begin()+n, minn);

}

void delAboveAverage (vctr& v)

{

float sum=0.0;

for (int i=0; i<v.size(); i++)

{

sum+=v[i];

}

sum/=v.size();

cout<<"Среднее = "<<sum<<'\n'<<"Элементы, больше среднего: ";

for (int i=0; i<v.size(); i++)

{

if (v[i]>sum)

{

cout<<v[i]<<' ';

v.erase (v.begin()+i);

}

}

}

void multiplyByMax(vctr& v)

{

float maxx=0.0;

for (int i=1; i<v.size(); i++)

{

if (v[i]>maxx) maxx=v[i];

}

cout<<"Максимальный элемент = "<<maxx<<'\n';

for (int i=0; i<v.size(); i++)

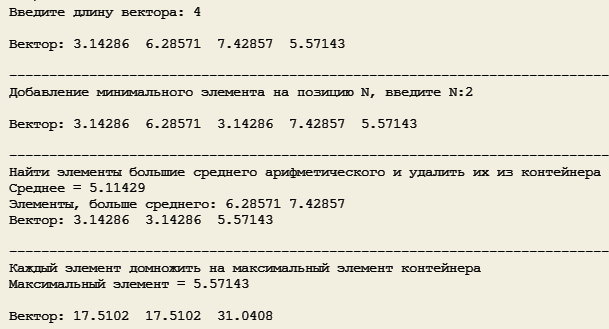
{

v[i]\*=maxx;

}

}

**Скриншоты результатов работы программы**

****

## 18.11.2

**Постановка задачи**

(Вариант 10)

Задача 2.

1.  Создать последовательный контейнер.

2.  Заполнить его элементами пользовательского типа(тип указан в варианте).

3.  Добавить элементы в соответствии с заданием

4.  Удалить элементы в соответствии с заданием.

5.  Выполнить задание варианта для полученного контейнера.

6.  Выполнение всех заданий оформить в виде глобальных функций.

Задача 1

1.  Контейнер - вектор

2.  Тип элементов - Money

Задание 3

Найти минимальный элемент и добавить его на заданную позицию

контейнера

Задание 4

Найти элементы большие среднего арифметического и

удалить их из контейнера

Задание 5

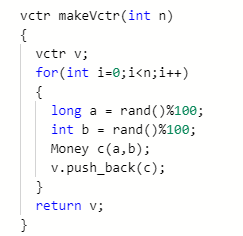
Каждый элемент домножить на максимальный элемент контейнера

**1.** Для решения задачи необходимо:

1.1.  Создать последовательный контейнер.

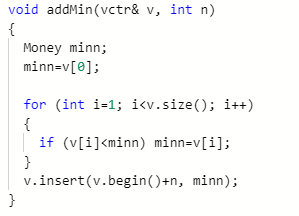


1.2.  Заполнить его элементами типаMoney.



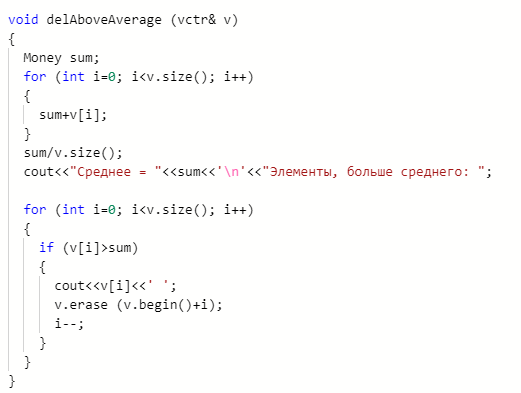
1.3.Найти минимальный элемент и добавить его на заданную позицию

контейнера .

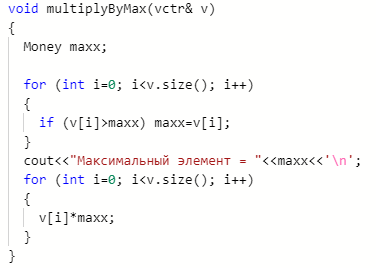


1.4. Найти элементы большие среднего арифметического и

удалить их из контейнера.



1.5. Каждый элемент домножить на максимальный элемент контейнера.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался тип Money.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Массив элементов был представлен в виде вектора, с типом элементов Money.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий действия над элементами вектора, используя ранее реализованные функции.



**Решение**

**Main.cpp**

#include<iostream>

#include<vector>

#include<cstdlib>

#include"Money.h"

usingnamespace std;

typedef vector<Money> vctr;

vctr makeVctr(int);

void printVctr(vctr);

void addMin(vctr&, int);

void delAboveAverage (vctr&);

void multiplyByMax(vctr&);

int main()

{

try

{

vctr v;

vctr::iterator vi=v.begin();

int n;

// 1, 2

cout<<"Введите длину вектора: ";

cin>>n;

if (n<1) throw0;

v=makeVctr(n);

printVctr(v);

//

cout<<"----------------------------------------------------------------------------\n";

// 3

cout<<"Добавление минимального элемента на позицию N, введите N:";

cin>>n;

if (n<1) throw0;

addMin(v, n);

printVctr(v);

//

cout<<"----------------------------------------------------------------------------\n";

//4

cout<<"Найти элементы большие среднего арифметического и удалить их из контейнера\n";

delAboveAverage(v);

printVctr(v);

//

cout<<"----------------------------------------------------------------------------\n";

// 5

cout<<"Каждый элемент домножить на максимальный элемент контейнера\n";

multiplyByMax(v);

printVctr(v);

//

}

catch(int)

{

cout<<"Ошибка";

}

}

vctr makeVctr(int n)

{

vctr v;

for(int i=0;i<n;i++)

{

long a = rand()%100;

int b = rand()%100;

Money c(a,b);

v.push\_back(c);

}

return v;

}

void printVctr(vctr v)

{

cout<<"\nВектор: ";

for (int i=0; i<v.size(); i++)

{

cout<<v[i]<<" ";

}

cout<<'\n';

cout<<endl;

}

void addMin(vctr& v, int n)

{

Money minn;

minn=v[0];

for (int i=1; i<v.size(); i++)

{

if (v[i]<minn) minn=v[i];

}

v.insert(v.begin()+n, minn);

}

void delAboveAverage (vctr& v)

{

Money sum;

for (int i=0; i<v.size(); i++)

{

sum+v[i];

}

sum/v.size();

cout<<"Среднее = "<<sum<<'\n'<<"Элементы, больше среднего: ";

for (int i=0; i<v.size(); i++)

{

if (v[i]>sum)

{

cout<<v[i]<<' ';

v.erase (v.begin()+i);

i--;

}

}

}

void multiplyByMax(vctr& v)

{

Money maxx;

for (int i=0; i<v.size(); i++)

{

if (v[i]>maxx) maxx=v[i];

}

cout<<"Максимальный элемент = "<<maxx<<'\n';

for (int i=0; i<v.size(); i++)

{

v[i]\*maxx;

}

}

**Money.h**

#pragma once

#include<iostream>

usingnamespace std;

class Money

{

  long rub;

  int kop;

public:

  Money()

  {

    rub = 0;

    kop = 0;

  }

  Money(long r, int k)

  {

    if (k<100)

    {

      rub = r;

      kop = k;

    }

    else

    {

      rub = r + k / 100;

      kop = k % 100;

    }

    if (r <0 || k <0)

    {

      cout <<"Ошибка, количество рублей и копеек должно быть > 0";

      rub = 0;

      kop = 0;

    }

  }

Money&operator--();

  Money operator--(int);

  Money&operator=(const Money&);

  booloperator==(Money&);

  booloperator==(int&);

booloperator!=(Money &);

  booloperator>(Money&);

  booloperator<(Money&);

Money&operator+(Money&);

Money&operator\*(Money&);

Money&operator/(int);

  ~Money() {}

friend istream&operator>> (istream&in, Money&m);

  friend ostream&operator<<(ostream&out, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

  kop = 0;

  return \*this;

}

Money Money::operator--(int)

{

  Money m(rub, kop);

  kop = 0;

  return m;

}

bool Money::operator==(Money &m)

{

  if (m.rub == rub && m.kop == kop)

  {

    returntrue;

  }

  else

  {

    returnfalse;

  }

}

bool Money::operator==(int&m)

{

  if (rub == m) returntrue;

  elsereturnfalse;

}

Money&Money::operator=(const Money&t)

{

if(&t==this) return \*this;

rub=t.rub;

kop=t.kop;

return \*this;

}

bool Money::operator!=(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returnfalse;

}

else

{

returntrue;

}

}

bool Money::operator>(Money&m)

{

  if (rub < m.rub)

  {

    returnfalse;

  }

  if (rub > m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop > m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

bool Money::operator<(Money&m)

{

  if (rub > m.rub)

  {

    returnfalse;

  }

  if (rub < m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop < m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

Money& Money::operator+(Money& m)

{

rub=rub+m.rub;

if (kop+m.kop>=100)

{

kop=kop+m.kop-100;

rub++;

}

else

{

kop=kop+m.kop;

}

return (\*this);

}

Money& Money::operator\*(Money&m)

{

rub=rub\*m.rub;

if (kop\*m.kop>=100)

{

kop=(kop\*m.kop)%100;

rub=rub+kop\*m.kop/100;

}

else

{

kop=kop\*m.kop;

}

return (\*this);

}

Money& Money::operator/(int k)

{

rub=rub/k;

kop=kop/k;

return (\*this);

}

istream&operator>> (istream&in, Money& m)

{

  cout <<"Рубли: ";

  in>> m.rub;

  cout <<"Копейки: ";

  in>> m.kop;

  returnin;

}

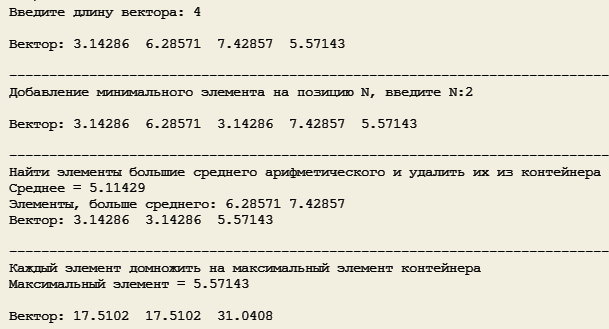
ostream&operator<<(ostream&out, const Money&m)

{

  return (out << m.rub <<","<< m.kop);

}

**Скриншоты результатов работы программы**

****

## 18.11.3

**Постановка задачи**

(Вариант 10)

Задача 3

1.  Создать параметризированный класс, используя в качестве контейнера последовательный контейнер.

2.  Заполнить его элементами.

3.  Добавить элементы в соответствии с заданием

4.  Удалить элементы в соответствии с заданием.

5.  Выполнить задание варианта для полученного контейнера.

6.  Выполнение всех заданий оформить в виде методов параметризированного класса.

Задача 3

Параметризированный класс – Вектор (см. лабораторную работу №7)

Задание 3

Найти минимальный элемент и добавить его на заданную позицию

контейнера

Задание 4

Найти элементы большие среднего арифметического и

удалить их из контейнера

Задание 5

Каждый элемент домножить на максимальный элемент контейнера

**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1. Создать параметризованный класс Vector.

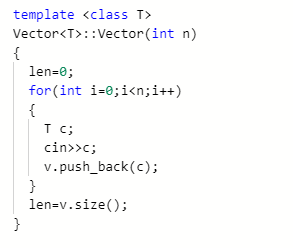


1.2.  Инстанцировать переменную типа Vector.





1.3.  Заполнить его элементами типаMoney.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался тип Money.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Массив элементов был представлен в виде вектора, с типом элементов Money.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий действия над элементами вектора, используя ранее реализованные функции.



**Решение**

**Main.cpp**

#include<iostream>

#include<vector>

#include<cstdlib>

#include"Vector.h"

#include"Money.h"

usingnamespace std;

typedef Vector<Money> vctr;

int main()

{

try

{

intn;

// 1, 2

cout<<"Введите длину вектора: ";

cin>>n;

if (n<1) throw0;

vctr v(n);

v.Print();

//

cout<<"----------------------------------------------------------------------------\n";

// 3

cout<<"Добавление минимального элемента на позицию N, введите N:";

cin>>n;

if (n<1 || n>v.len) throw0;

v.addMin(n);

v.Print();

//

cout<<"----------------------------------------------------------------------------\n";

//4

cout<<"Найти элементы большие среднего арифметического и удалить их из контейнера\n";

v.delAboveAverage();

v.Print();

//

cout<<"----------------------------------------------------------------------------\n";

// 5

cout<<"Каждый элемент домножить на максимальный элемент контейнера\n";

v.multiplyByMax();

v.Print();

//

}

catch(int)

{

cout<<"Ошибка";

}

}

**Vector.h**

#pragma once

#include<vector>

#include<iostream>

usingnamespace std;

template<class T>

class Vector

{

vector <T> v;

public:

int len;

Vector();

Vector(int n);

void Print();

~Vector(void);

void addMin(int);

void delAboveAverage ();

void multiplyByMax();

};

template<class T>

Vector<T>::Vector()

{

len=0;

}

template<class T>

Vector<T>::Vector(int n)

{

len=0;

for(int i=0;i<n;i++)

{

T c;

cin>>c;

v.push\_back(c);

}

len=v.size();

}

template<class T>

Vector<T>::~Vector(void)

{}

template<class T>

void Vector<T>::Print()

{

cout<<"\nВектор: ";

for(int i=0; i<v.size(); i++) cout<<v[i]<<"  ";

cout<<'\n';

}

template<class T>

void Vector<T>::addMin(int n)

{

T minn;

minn=v[0];

for (int i=1; i<v.size(); i++)

{

if (v[i]<minn) minn=v[i];

}

v.insert(v.begin()+n, minn);

}

template<class T>

void Vector<T>::delAboveAverage()

{

T sum;

for (int i=0; i<v.size(); i++)

{

sum+v[i];

}

sum/v.size();

cout<<"Среднее = "<<sum<<'\n'<<"Элементы, больше среднего: ";

for (int i=0; i<v.size(); i++)

{

if (v[i]>sum)

{

cout<<v[i]<<' ';

v.erase (v.begin()+i);

i--;

}

}

}

template<class T>

void Vector<T>::multiplyByMax()

{

T maxx;

for (int i=1; i<v.size(); i++)

{

if (v[i]>maxx) maxx=v[i];

}

cout<<"Максимальный элемент = "<<maxx<<'\n';

for (int i=0; i<v.size(); i++)

{

v[i]\*maxx;

}

}

**Money.h**

#pragma once

#include<iostream>

usingnamespace std;

class Money

{

  long rub;

  int kop;

public:

  Money()

  {

    rub = 0;

    kop = 0;

  }

  Money(long r, int k)

  {

    if (k<100)

    {

      rub = r;

      kop = k;

    }

    else

    {

      rub = r + k / 100;

      kop = k % 100;

    }

    if (r <0 || k <0)

    {

      cout <<"Ошибка, количество рублей и копеек должно быть > 0";

      rub = 0;

      kop = 0;

    }

  }

Money&operator--();

  Money operator--(int);

  Money&operator=(const Money&);

  booloperator==(Money&);

  booloperator==(int&);

booloperator!=(Money &);

  booloperator>(Money&);

  booloperator<(Money&);

Money&operator+(Money&);

Money&operator\*(Money&);

Money&operator/(int);

  ~Money() {}

friend istream&operator>> (istream&in, Money&m);

  friend ostream&operator<<(ostream&out, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

  kop = 0;

  return \*this;

}

Money Money::operator--(int)

{

  Money m(rub, kop);

  kop = 0;

  return m;

}

bool Money::operator==(Money &m)

{

  if (m.rub == rub && m.kop == kop)

  {

    returntrue;

  }

  else

  {

    returnfalse;

  }

}

bool Money::operator==(int&m)

{

  if (rub == m) returntrue;

  elsereturnfalse;

}

Money&Money::operator=(const Money&t)

{

if(&t==this) return \*this;

rub=t.rub;

kop=t.kop;

return \*this;

}

bool Money::operator!=(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returnfalse;

}

else

{

returntrue;

}

}

bool Money::operator>(Money&m)

{

  if (rub < m.rub)

  {

    returnfalse;

  }

  if (rub > m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop > m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

bool Money::operator<(Money&m)

{

  if (rub > m.rub)

  {

    returnfalse;

  }

  if (rub < m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop < m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

Money& Money::operator+(Money& m)

{

rub=rub+m.rub;

if (kop+m.kop>=100)

{

kop=kop+m.kop-100;

rub++;

}

else

{

kop=kop+m.kop;

}

return (\*this);

}

Money& Money::operator\*(Money&m)

{

rub=rub\*m.rub;

if (kop\*m.kop>=100)

{

rub=rub+kop\*m.kop/100;

kop=(kop\*m.kop)%100;

}

else

{

kop=kop\*m.kop;

}

return (\*this);

}

Money& Money::operator/(int k)

{

rub=rub/k;

kop=kop/k;

return (\*this);

}

istream&operator>> (istream&in, Money& m)

{

  cout <<"Рубли: ";

  in>> m.rub;

  cout <<"Копейки: ";

  in>> m.kop;

  returnin;

}

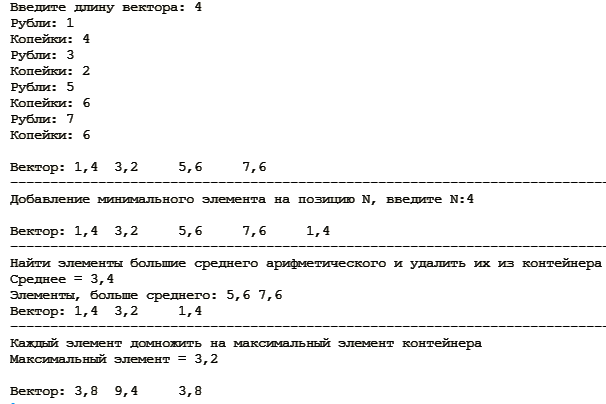
ostream&operator<<(ostream&out, const Money&m)

{

  return (out << m.rub <<","<< m.kop);

}

**Скриншоты результатов работы программы**

****

## 18.11.4

**Постановка задачи**

(Вариант 10)

Задача 4

1.  Создать адаптер контейнера.

2.  Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3.  Добавить элементы в соответствии с заданием

4.  Удалить элементы в соответствии с заданием.

5.  Выполнить задание варианта для полученного контейнера.

6.  Выполнение всех заданий оформить в виде глобальных функций.

Задача 4

Адаптер контейнера – очередь.

Задание 3

Найти минимальный элемент и добавить его на заданную позицию

контейнера

Задание 4

Найти элементы большие среднего арифметического и

удалить их из контейнера

Задание 5

Каждый элемент домножить на максимальный элемент контейнера

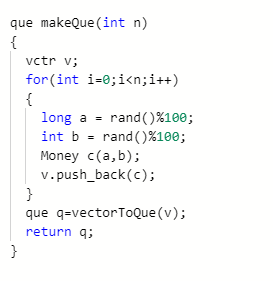
**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1.  Создать адаптер контейнера.

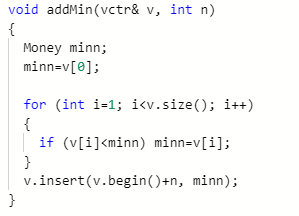


1.2.  Заполнить его элементами типа Money.



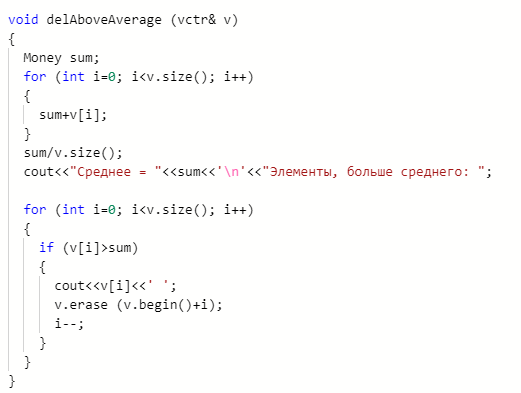
1.3.Найти минимальный элемент и добавить его на заданную позицию

контейнера .

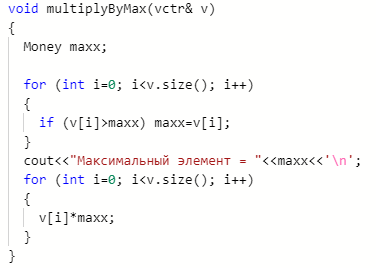


1.4. Найти элементы большие среднего арифметического и

удалить их из контейнера.



1.5. Каждый элемент домножить на максимальный элемент контейнера.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался тип Money.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Массив элементов был представлен в виде очереди, с типом элементов Money.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код для выполнения поставленных заданий.



**Решение**

**Main.cpp**

#include<iostream>

#include<vector>

#include<queue>

#include<cstdlib>

#include<vector>

#include"Money.h"

usingnamespace std;

typedef vector<Money> vctr;

typedef queue<Money> que;

vctr queToVector(que);

que vectorToQue(vctr);

que makeQue(int);

void printQue(que);

void addMin(vctr&, int);

void delAboveAverage (vctr&);

void multiplyByMax(vctr&);

int main()

{

try

{

vctr v;

que q;

int n;

// 1, 2

cout<<"Введите длину очереди: ";

cin>>n;

if (n<1) throw0;

q=makeQue(n);

printQue(q);

//

cout<<"----------------------------------------------------------------------------\n";

// 3

cout<<"Добавление минимального элемента на позицию N, введите N:";

cin>>n;

if (n<1) throw0;

v=queToVector(q);

addMin(v, n);

q=vectorToQue(v);

printQue(q);

//

cout<<"----------------------------------------------------------------------------\n";

//4

cout<<"Найти элементы большие среднего арифметического и удалить их из контейнера\n";

v=queToVector(q);

delAboveAverage(v);

q=vectorToQue(v);

printQue(q);

//

cout<<"----------------------------------------------------------------------------\n";

// 5

cout<<"Каждый элемент домножить на максимальный элемент контейнера\n";

v=queToVector(q);

multiplyByMax(v);

q=vectorToQue(v);

printQue(q);

//

}

catch(int)

{

cout<<"Ошибка";

}

}

que makeQue(int n)

{

vctr v;

for(int i=0;i<n;i++)

{

long a = rand()%100;

int b = rand()%100;

Money c(a,b);

v.push\_back(c);

}

que q=vectorToQue(v);

return q;

}

void printQue(que q)

{

vctr v=queToVector(q);

cout<<"\nОчередь: ";

for (int i=0; i<v.size(); i++)

{

cout<<v[i]<<" ";

}

cout<<'\n';

cout<<endl;

}

void addMin(vctr& v, int n)

{

Money minn;

minn=v[0];

for (int i=1; i<v.size(); i++)

{

if (v[i]<minn) minn=v[i];

}

v.insert(v.begin()+n, minn);

}

void delAboveAverage (vctr& v)

{

Money sum;

for (int i=0; i<v.size(); i++)

{

sum+v[i];

}

sum/v.size();

cout<<"Среднее = "<<sum<<'\n'<<"Элементы, больше среднего: ";

for (int i=0; i<v.size(); i++)

{

if (v[i]>sum)

{

cout<<v[i]<<' ';

v.erase (v.begin()+i);

i--;

}

}

}

void multiplyByMax(vctr& v)

{

Money maxx;

for (int i=0; i<v.size(); i++)

{

if (v[i]>maxx) maxx=v[i];

}

cout<<"Максимальный элемент = "<<maxx<<'\n';

for (int i=0; i<v.size(); i++)

{

v[i]\*maxx;

}

}

vctr queToVector(que q)

{

vctr v;

while(!q.empty())

{

v.push\_back(q.front());

q.pop();

}

return v;

}

que vectorToQue(vctr v)

{

que q;

for(int i=0;i<v.size();i++)

{

q.push(v[i]);

}

return q;

}

**Money.h**

#pragma once

#include<iostream>

usingnamespace std;

class Money

{

  long rub;

  int kop;

public:

  Money()

  {

    rub = 0;

    kop = 0;

  }

  Money(long r, int k)

  {

    if (k<100)

    {

      rub = r;

      kop = k;

    }

    else

    {

      rub = r + k / 100;

      kop = k % 100;

    }

    if (r <0 || k <0)

    {

      cout <<"Ошибка, количество рублей и копеек должно быть > 0";

      rub = 0;

      kop = 0;

    }

  }

Money&operator--();

  Money operator--(int);

  Money&operator=(const Money&);

  booloperator==(Money&);

  booloperator==(int&);

booloperator!=(Money &);

  booloperator>(Money&);

  booloperator<(Money&);

Money&operator+(Money&);

Money&operator\*(Money&);

Money&operator/(int);

  ~Money() {}

friend istream&operator>> (istream&in, Money&m);

  friend ostream&operator<<(ostream&out, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

  kop = 0;

  return \*this;

}

Money Money::operator--(int)

{

  Money m(rub, kop);

  kop = 0;

  return m;

}

bool Money::operator==(Money &m)

{

  if (m.rub == rub && m.kop == kop)

  {

    returntrue;

  }

  else

  {

    returnfalse;

  }

}

bool Money::operator==(int&m)

{

  if (rub == m) returntrue;

  elsereturnfalse;

}

Money&Money::operator=(const Money&t)

{

if(&t==this) return \*this;

rub=t.rub;

kop=t.kop;

return \*this;

}

bool Money::operator!=(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returnfalse;

}

else

{

returntrue;

}

}

bool Money::operator>(Money&m)

{

  if (rub < m.rub)

  {

    returnfalse;

  }

  if (rub > m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop > m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

bool Money::operator<(Money&m)

{

  if (rub > m.rub)

  {

    returnfalse;

  }

  if (rub < m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop < m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

Money& Money::operator+(Money& m)

{

rub=rub+m.rub;

if (kop+m.kop>=100)

{

kop=kop+m.kop-100;

rub++;

}

else

{

kop=kop+m.kop;

}

return (\*this);

}

Money& Money::operator\*(Money&m)

{

rub=rub\*m.rub;

if (kop\*m.kop>=100)

{

rub=rub+kop\*m.kop/100;

kop=(kop\*m.kop)%100;

}

else

{

kop=kop\*m.kop;

}

return (\*this);

}

Money& Money::operator/(int k)

{

rub=rub/k;

kop=kop/k;

return (\*this);

}

istream&operator>> (istream&in, Money& m)

{

  cout <<"Рубли: ";

  in>> m.rub;

  cout <<"Копейки: ";

  in>> m.kop;

  returnin;

}

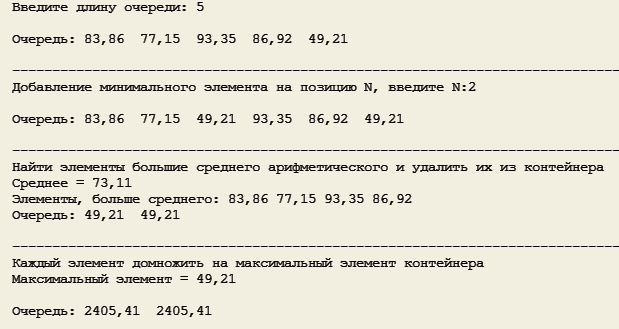
ostream&operator<<(ostream&out, const Money&m)

{

  return (out << m.rub <<","<< m.kop);

}

**Скриншоты результатов работы программы**

****

## 18.11.5

**Постановка задачи**

(Вариант 10)

**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1. Создать параметризованный класс Vector.

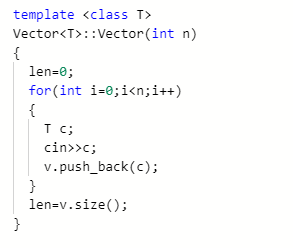


1.2.  Инстанцировать переменную типа Vector.





1.3.  Заполнить его элементами типаMoney.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался тип Money.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Массив элементов был представлен в виде вектора, с типом элементов Money.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий действия над элементами вектора, используя ранее реализованные функции.



**Решение**

**Main.cpp**

#include<iostream>

#include<vector>

#include<queue>

#include<cstdlib>

#include"Vector.h"

#include"Money.h"

usingnamespace std;

typedef Vector<Money> vctr;

typedef queue<Money> que;

queue<Money> makeQue(int);

int main()

{

try

{

int n;

// 1, 2

cout<<"Введите длину очереди ";

cin>>n;

if (n<1) throw0;

vctr v;

que q;

q=makeQue(n);

v.queToVector(q, n);

v.Print();

//

cout<<"----------------------------------------------------------------------------\n";

// 3

cout<<"Добавление минимального элемента на позицию N, введите N:";

cin>>n;

if (n<1 || n>v.len) throw0;

v.addMin(n);

v.Print();

//

cout<<"----------------------------------------------------------------------------\n";

//4

cout<<"Найти элементы большие среднего арифметического и удалить их из контейнера\n";

v.delAboveAverage();

v.Print();

//

cout<<"----------------------------------------------------------------------------\n";

// 5

cout<<"Каждый элемент домножить на максимальный элемент контейнера\n";

v.multiplyByMax();

v.Print();

//

}

catch(int)

{

cout<<"Ошибка";

}

}

que makeQue(int n)

{

vctr v;

for(int i=0;i<n;i++)

{

long a = rand()%100;

int b = rand()%100;

Money c(a,b);

v.push(c);

}

que q=v.vectorToQue();

return q;

}

**Vector.h**

#pragma once

#include<vector>

#include<iostream>

usingnamespace std;

template<class T>

class Vector

{

vector <T> v;

public:

int len;

Vector();

Vector(int n);

void Print();

~Vector(void);

void push(T);

void addMin(int);

void delAboveAverage ();

void multiplyByMax();

void queToVector(queue<T>, int);

queue<T> vectorToQue();

};

template<class T>

Vector<T>::Vector()

{

len=0;

}

template<class T>

Vector<T>::Vector(int n)

{

len=0;

for(int i=0;i<n;i++)

{

T c;

cin>>c;

v.push\_back(c);

}

len=v.size();

}

template<class T>

Vector<T>::~Vector(void)

{}

template<class T>

void Vector<T>::push(T arg)

{

v.push\_back(arg);

}

template<class T>

void Vector<T>::Print()

{

cout<<"\nВектор: ";

for(int i=0; i<v.size(); i++) cout<<v[i]<<"  ";

cout<<'\n';

}

template<class T>

void Vector<T>::addMin(int n)

{

T minn;

minn=v[0];

for (int i=1; i<v.size(); i++)

{

if (v[i]<minn) minn=v[i];

}

v.insert(v.begin()+n, minn);

}

template<class T>

void Vector<T>::delAboveAverage()

{

T sum;

for (int i=0; i<v.size(); i++)

{

sum+v[i];

}

sum/v.size();

cout<<"Среднее = "<<sum<<'\n'<<"Элементы, больше среднего: ";

for (int i=0; i<v.size(); i++)

{

if (v[i]>sum)

{

cout<<v[i]<<' ';

v.erase (v.begin()+i);

i--;

}

}

}

template<class T>

void Vector<T>::multiplyByMax()

{

T maxx;

for (int i=1; i<v.size(); i++)

{

if (v[i]>maxx) maxx=v[i];

}

cout<<"Максимальный элемент = "<<maxx<<'\n';

for (int i=0; i<v.size(); i++)

{

v[i]\*maxx;

}

}

template<class T>

void Vector<T>::queToVector(queue<T> q, int n)

{

while(!q.empty())

{

v.push\_back(q.front());

q.pop();

}

(\*this).len=n;

}

template<class T>

queue<T> Vector<T>::vectorToQue()

{

queue<T> q;

for(int i=0;i<v.size();i++)

{

q.push(v[i]);

}

return q;

}

**Money.h**

#pragma once

#include<iostream>

usingnamespace std;

class Money

{

  long rub;

  int kop;

public:

  Money()

  {

    rub = 0;

    kop = 0;

  }

  Money(long r, int k)

  {

    if (k<100)

    {

      rub = r;

      kop = k;

    }

    else

    {

      rub = r + k / 100;

      kop = k % 100;

    }

    if (r <0 || k <0)

    {

      cout <<"Ошибка, количество рублей и копеек должно быть > 0";

      rub = 0;

      kop = 0;

    }

  }

Money&operator--();

  Money operator--(int);

  Money&operator=(const Money&);

  booloperator==(Money&);

  booloperator==(int&);

booloperator!=(Money &);

  booloperator>(Money&);

  booloperator<(Money&);

Money&operator+(Money&);

Money&operator\*(Money&);

Money&operator/(int);

  ~Money() {}

friend istream&operator>> (istream&in, Money&m);

  friend ostream&operator<<(ostream&out, const Money&m);

};

**Money.cpp**

#include"Money.h"

#include<iostream>

usingnamespace std;

Money &Money::operator--()

{

  kop = 0;

  return \*this;

}

Money Money::operator--(int)

{

  Money m(rub, kop);

  kop = 0;

  return m;

}

bool Money::operator==(Money &m)

{

  if (m.rub == rub && m.kop == kop)

  {

    returntrue;

  }

  else

  {

    returnfalse;

  }

}

bool Money::operator==(int&m)

{

  if (rub == m) returntrue;

  elsereturnfalse;

}

Money&Money::operator=(const Money&t)

{

if(&t==this) return \*this;

rub=t.rub;

kop=t.kop;

return \*this;

}

bool Money::operator!=(Money &m)

{

if (m.rub==rub && m.kop==kop)

{

returnfalse;

}

else

{

returntrue;

}

}

bool Money::operator>(Money&m)

{

  if (rub < m.rub)

  {

    returnfalse;

  }

  if (rub > m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop > m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

bool Money::operator<(Money&m)

{

  if (rub > m.rub)

  {

    returnfalse;

  }

  if (rub < m.rub)

  {

    returntrue;

  }

  if (rub == m.rub)

  {

    if (kop < m.kop)

    {

      returntrue;

    }

    else

    {

      returnfalse;

    }

  }

returnfalse;

}

Money& Money::operator+(Money& m)

{

rub=rub+m.rub;

if (kop+m.kop>=100)

{

kop=kop+m.kop-100;

rub++;

}

else

{

kop=kop+m.kop;

}

return (\*this);

}

Money& Money::operator\*(Money&m)

{

rub=rub\*m.rub;

if (kop\*m.kop>=100)

{

rub=rub+kop\*m.kop/100;

kop=(kop\*m.kop)%100;

}

else

{

kop=kop\*m.kop;

}

return (\*this);

}

Money& Money::operator/(int k)

{

rub=rub/k;

kop=kop/k;

return (\*this);

}

istream&operator>> (istream&in, Money& m)

{

  cout <<"Рубли: ";

  in>> m.rub;

  cout <<"Копейки: ";

  in>> m.kop;

  returnin;

}

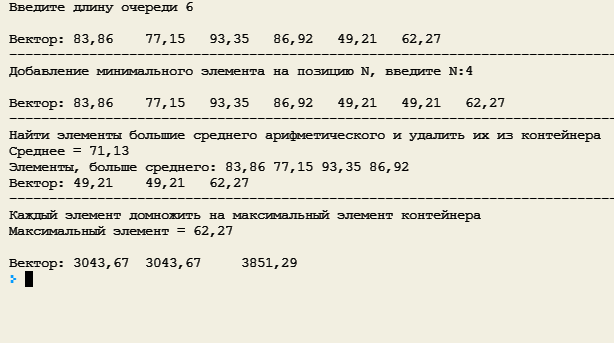
ostream&operator<<(ostream&out, const Money&m)

{

  return (out << m.rub <<","<< m.kop);

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Из каких частей состоит библиотека STL?**

Стандартная библиотека шаблонов состоит из двух основных частей: набора контейнерных классов и набора обобщенных алгоритмов.

**Какие типы контейнеров существуют в STL?**

Контейнеры STL можно разделить на два типа: последовательные и ассоциативные.

**Что нужно сделать для использования контейнера STL в своей программе?**

Для использования контейнера в программе необходимо включить в нее соответствующий заголовочный файл. Тип объектов, сохраняемых в контейнере, задается с помощью аргумента шаблона, например:

#include <vector> #include <list> #include “person.h”

….. vector<int> v; list<person> l;

**Что представляет собой итератор?**

Итератор - более общее понятие, чем указатель. Позволяет последовательно перемещаться по контейнеру.

**Какие операции можно выполнять над итераторами?**

К основным операциям, выполняемым с любыми итераторами, относятся:

Разыменование итератора: если р — итератор, то \*р — значение объекта, на который он ссылается.

Присваивание одного итератора другому.

Сравнение итераторов на равенство и неравенство (== и !=).

Перемещение его по всем элементам контейнера с помощью префиксного (++р) или постфиксного (р++) инкремента.

**Каким образом можно организовать цикл для перебора контейнера с использованием итератора?**

for (i = first; i != last; ++i),

**Какие типы итераторов существуют?**

В STL существуют следующие типы итераторов:

входные,

выходные,

прямые,

двунаправленные итераторы,

итераторы произвольного доступа.

**Перечислить операции и методы общие для всех контейнеров.**

|  |  |
| --- | --- |
| **Операция или метод** | **Пояснение** |
| Операции равенства (==) и  неравенства (!=) | Возвращают значение true или false |
| Операция присваивания (=) | Копирует один контейнер в другой |
| clear | Удаляет все элементы |
| insert | Добавляет один элемент или диапазон элементов |
| erase | Удаляет один элемент или диапазон элементов |
| size\_type size() const | Возвращает число элементов |
| size\_type max\_size() const | Возвращает максимально допустимый размер  контейнера |
| bool empty0 const | Возвращает true, если контейнер пуст |
| iterator begin() | Возвращают итератор на начало контейнера (итерации будут производиться в прямом  направлении) |
| iterator end() | Возвращают итератор на конец контейнера  (итерации в прямом направлении будут закончены) |
| reverse\_iterator begin() | Возвращают реверсивный итератор на конец  контейнера (итерации будут производиться в обратном направлении) |
| reverse\_iterator end() | Возвращают реверсивный итератор на начало  контейнера (итерации в обратном направлении будут закончены |

**Какие операции являются эффективными для контейнера vector? Почему?**

Вставка в конец вектора, так как нет необходимости сдвигать все элементы вектора.

**Какие операции являются эффективными для контейнера list? Почему?**

Вставка и удаление на любую позицию, так как достаточно лишь изменить указатели предыдущего и следующего элементов.

**Какие операции являются эффективными для контейнера deque? Почему?**

Вставка и удаление в конец и в начало, так как, в отличие от вектора, не надо сдвигать все элементы.

**Перечислить методы, которые поддерживает последовательный контейнер vector.**

|  |  |
| --- | --- |
| Vector | |
| push\_back() | добавление  в конец |
| pop\_back() | удаление из  конца |
| insert | Вставка в  произвольн ое место |
| erase | удаление из произвольн  ог оместа |
| []  at | доступ к произвольн ому  элементу |
|  |  |
|  |  |
| swap | обмен векторов |
| clear() | очистить  вектор |
|  |  |

**Перечислить методы, которые поддерживает последовательный контейнер list.**

|  |  |
| --- | --- |
| List | |
| push\_back(T&key) | добавление  в конец |
| pop\_back() | удаление из  конца |
| push\_front(T&key) | добавление в начало |
| pop\_front() | удаление из начала |
| insert | Вставка в произвольн ое место |
| erase | удаление из произвольн  ог оместа |
|  |  |
| swap | Обмен списков |
| clear() | очистить  вектор |
| splice | сцепка  списков |

**Перечислить методы, которые поддерживает последовательный контейнер deque.**

|  |  |
| --- | --- |
| Deque | |
| push\_back(T&key) | добавление  в конец |
| pop\_back() | удаление из  конца |
| push\_front(T&key) | добавление в начало |
| pop\_front() | удаление из начала |
| insert | Вставка в произвольн ое место |
| erase | удаление из произвольн  ог оместа |
| []  at | доступ к произвольн ому  элементу |

**Задан контейнер vector. Как удалить из него элементы со 2 по 5?**

Vector v(5);

For (int i=0; i<v.size(); i++) v.erase(v.begin()+i);

**Задан контейнер vector. Как удалить из него последний элемент?**

Vector v(5);

v.erase(v.end()-1);

**Задан контейнер list. Как удалить из него элементы со 2 по 5?**

List l;

For (int i=0; i<4; i++) l.pop\_back();

**Задан контейнер list. Как удалить из него последний элемент?**

List l;

l.pop\_back();

**Задан контейнер deque. Как удалить из него элементы со 2 по 5?**

deque dq;

For (int i=0; i<4; i++) dq.pop\_back();

**Задан контейнер deque. Как удалить из него последний элемент?**

deque dq;

dq.pop\_back();

**Написать функцию для печати последовательного контейнера с использованием итератора.**

Vector v(10);

Iterator::vector<int> I;

For (i=v.begin(); i<v.end(); i++) cout<<(\*i)

**Что представляют собой адаптеры контейнеров?**

Специализированные последовательные контейнеры — стек, очередь и очередь с приоритетами — не являются самостоятельными контейнерными классами, а реализованы на основе рассмотренных выше классов, поэтому они называются адаптерами контейнеров.

**Чем отличаются друг от друга объявления stack<int> s и stack<int, list<int> > s?**

Первое – создает стек на базе двунаправленной очереди, второе – на базе списка.

Смысл такой реализации заключается в том, что специализированный класс просто переопределяет интерфейс класса-прототипа, ограничивая его только теми методами, которые нужны новому классу. Cтек не позволяет выполнить произвольный доступ к своим элементам, а также не дает возможности пошагового перемещения, т. е. итераторы в стеке не поддерживаются.

**Перечислить методы, которые поддерживает контейнер stack.**

Методы класса stack:

push () - добавление в конец;

pop () - удаление из конца;

top () - получение текущего элемента стека;

empty() - проверка пустой стек или нет;

size () – получение размера стека.

**Перечислить методы, которые поддерживает контейнер queue.**

Методы класса queue:

push () – добавление в конец очереди;

pop () – удаление из начала очереди;

front () – получение первого элемента очереди;

back()- получение последнего элемента очереди;

empty () – проверка пустая очередь или нет;

size() – получение размера очереди.

**Чем отличаются друг от друга контейнеры queue и priority\_queue?**

Очередь с приоритетами отличается от обычной очереди тем, что для извлечения выбирается максимальный элемент из хранимых в контейнере. Поэтому после каждого изменения состояния очереди максимальный элемент из оставшихся сдвигается в начало контейнера.

**Задан контейнер stack. Как удалить из него элемент с заданным номером?**

#include <iostream>

#include <stack>

using namespace std;

int main()

{

stack <int> st;

stack <int> stNew;

int i, length= rand()%15+1, arr[50];

bool f=false;

for (i=0; i<length; i++)

{

st.push (rand()%100);

}

cout<<"Ваш стек: ";

for (i=length-1; i>-1; i--)

{

cout<<st.top()<<' ';

arr[i]=st.top();

st.pop();

}

int delPos;

cout<<"\nВведите позицию, с которой вы хотите удалить элемент ";

cin>>delPos;

for (i=0; i<length; i++)

{

if (i==length-delPos && !f)

{

arr[i]=-1;

f=true;

}

}

cout<<'\n';

for (i=0; i<length; i++)

{

stNew.push (arr[i]);

}

cout<<"Ваш стек: ";

for (i=length-1; i>-1; i--)

{

if (stNew.top()!=-1) cout<<stNew.top()<<' ';

arr[i]=stNew.top();

stNew.pop();

}

}

**Задан контейнер queue. Как удалить из него элемент с заданным номером?**

#include <iostream>

#include <queue>3

using namespace std;

int main ()

{

queue <int> qu;

queue <int> quNew;

int help, i, length = rand()%15+1, arr[50];

for (i=0; i<length; i++)

{

qu.push (rand()%100);

}

cout<<"Ваша очередь: ";

for (i=0; i<length; i++)

{

arr[i]=qu.front();

cout<<qu.front()<<' ';

qu.pop();

}

for (i=0; i<length; i++)

{

qu.push(arr[i]);

}

int delPos;

cout<<"\nВведите позицию, с которой вы хотите удалить элемент ";

cin>>delPos;

for (i=0; i<length; i++)

{

if (i!=delPos-1) quNew.push(qu.front());

qu.pop();

}

cout<<"\nВаша очередь: ";

for (i=0; i<(length-1); i++)

{

arr[i]=quNew.front();

cout<<quNew.front()<<' ';

quNew.pop();

}

}

**Написать функцию для печати контейнера stack с использованием итератора.**

*Очень интересно, как я должен применить итератор к стеку, если в этой же методичке написано, что нельзя там его использовать. Надеюсь мне этот вопрос на экзамене попадется.*

**Написать функцию для печати контейнера queue с использованием итератора.**

*А да? Ну тут можно не продолжать.*

## 18.12.1

**Постановка задачи**

(Вариант 10)

Задача 1.

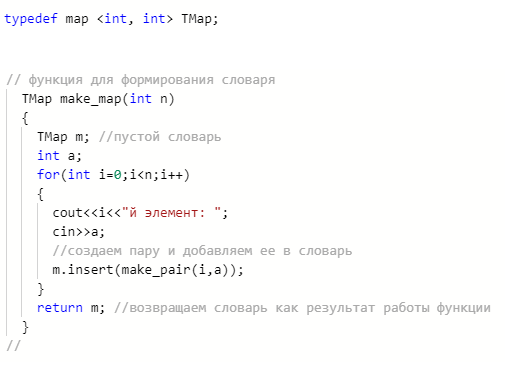
Задача 1

1. Контейнер–словарь(map);
2. типэлементов-int;
3. Найтисреднееарифметическоеэлементовсловаряидобавитьеговсловарь.
4. Удалитьмаксимальныйэлементизвектора.
5. Каждыйэлементразделитьнаминимальноезначениевектора.

**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1. Создать словарь и заполнить его элеентами типа int



1.2.Реализовать функции для выполнения задания.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов словаря и их индексов использоваля тип int.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Список элементов был представлен в виде словаря.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий поставленные задачи, применяя стандартные алгоритмы и реализованные функции.



**Решение**

**Main.cpp**

#include<iostream>

#include<map>

usingnamespace std;

typedefmap<int, int>TMap; // определяемтипдляработысословарем

typedefTMap::iteratorit; // итератор

TMapmake\_map (int); // функциядляформированиясловаря

void print\_map (TMap); // функция для печати словаря

int srednee (TMap); // функция для вычисления среднего арифметического

int Max (TMap); // функция для поиска максимального элемента

int Min(TMap); // функиця поиска минимального элемента

void delenie(TMap&); // функция деления каждого элемента массива на минимальный элемент

// основная функция

int main()

{

int n;

// Создание и вывод массива

cout<<"Введите количество элементов: ";

cin>>n;

TMap m=make\_map(n); //создать словарь

print\_map(m); //напечатать словарь

//

cout<<"----------------------------------------------------------------------------\n";

// Вычисление среднего арифметического и добавление его в конец массива

//вычисление среднего

int el=srednee(m);

cout<<"Среднее арифметическое элементов массива = "<<el<<endl;

//добавлениевконец

m.insert(make\_pair(n,el));

print\_map(m);

//

cout<<"----------------------------------------------------------------------------\n";

// Поиск и удаление макс элемента

int max=Max(m);

cout<<"max: "<<m[max]<<" Его индекс в массиве: "<<max<<endl;

m.erase(max); //удалениеэлемента

print\_map(m);

//

cout<<"----------------------------------------------------------------------------\n";

// Поиск и деление каждого элемента массива на минимальный элемент

int min=Min(m);

cout<<"min: "<<m[min]<<" Егоиндексвмассиве: "<<min<<endl;

delenie(m);

print\_map(m);

//

}

// функция для формирования словаря

TMapmake\_map(intn)

{

TMap m; //пустой словарь

int a;

for(int i=0;i<n;i++)

{

cout<<i<<"йэлемент: ";

cin>>a;

//создаем пару и добавляем ее в словарь

m.insert(make\_pair(i,a));

}

return m; //возвращаем словарь как результат работы функции

}

//

// функция для печати словаря

void print\_map(TMap m)

{

cout<<"\nВашмассив:\n";

for(int i=0;i<m.size();i++) cout<<i<<": "<<m[i]<<" "<<endl;

}

//

// функция для вычисления среднего арифметического

int srednee(TMap v)

{

int s=0;

//переборсловаря

for(int i=0;i<v.size();i++) s+=v[i];

int n=v.size(); //количество элементов в словаре

return s/n;

}

//

// функция для поиска максимального элемента

int Max(TMap v)

{

it i=v.begin();

intnom=0; //номермаксимального

int k=0; //счетчик элементов

int m=(\*i).second; //значение первого элемента

while(i!=v.end())

{

if(m<(\*i).second)

{

m=(\*i).second; nom=k;

}

i++; //итератор

k++; //счетчик элементов

}

return nom; //номер max

}

//

// функиця поиска минимального элемента

int Min(TMap v)

{

it i=v.begin();

intnom=0,k=0;

int m=(\*i).second; //значение первого элемента

while(i!=v.end())

{

if(m>(\*i).second)

{

m=(\*i).second; nom=k;

}

i++; //итератор

k++; //счетчик элементов

}

return nom; //номер max

}

//

// функция деления каждого элемента массива на минимальный элемент

void delenie(TMap &v)

{

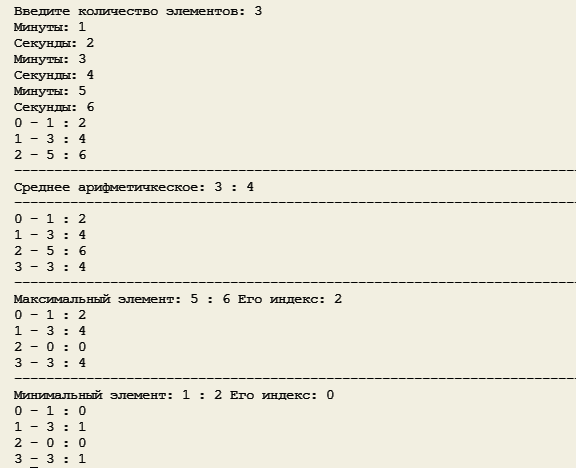
int m=v[Min(v)]; //значение минимального элемента

for(int i=0;i<v.size();i++) v[i]=v[i]/m;

}

//

**Скриншоты результатов работы программы**

****

## 18.12.2

**Постановка задачи**

(Вариант 10)

Задача 2.

1 Контейнер-словарь.

2 ТипэлементовTime.

3 Найтисреднееарифметическоесловаряи добавитьегов словарьподномером k.

4 Удалитьмаксимальныйэлементизсловаря.

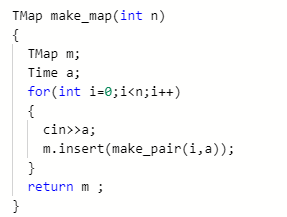
5 Каждыйэлементразделитьнаминимальноезначениесловаря.

6 Выполнениевсехзаданийоформитьввидеглобальныхфункций.

**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1. Создать словарь и заполнить его элеентами типаTime.



1.2.Реализовать функции для выполнения задания.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов словаря и их индексов использоваля тип Time.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Список элементов был представлен в виде словаря.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий поставленные задачи, применяя стандартные алгоритмы и реализованные функции.



**Решение**

**Main.cpp**

/\*

Ассоциативные контейнеры

Пользовательский тип

\*/

#include<iostream>

#include<map>

#include"Time.h"

usingnamespace std;

typedef map <int, Time> TMap;

typedef TMap::iterator it;

TMap make\_map(int n)

{

TMap m;

Time a;

for(int i=0;i<n;i++)

{

cin>>a;

m.insert(make\_pair(i,a));

}

return m ;

}

void print\_map(TMap m)

{

for(int i=0;i<m.size();i++) cout<<i<<" - "<<m[i]<<" "<<endl;

}

Time srednee(TMap m)

{

Time s=m[0];

for(int i=1;i<m.size();i++) s=s+m[i];

int n=m.size();

return s/n;

}

int Max(TMap v)

{

it i=v.begin();

int nom=0,k=0;

Time m=(\*i).second;

while(i!=v.end())

{

if(m < (\*i).second)

{

m=(\*i).second;

nom=k;

}

i++;

k++;

}

return nom;

}

int Min(TMap v)

{

it i=v.begin();

int nom=0,k=0;

Time m=(\*i).second;

while(i!=v.end())

{

if(m > (\*i).second)

{

m=(\*i).second;

nom=k;

}

i++;

k++;

}

return nom;

}

void delenie(TMap &v)

{

Time m=v[Min(v)];

for(int i=0;i<v.size();i++) v[i]=v[i]/m;

}

int main()

{

int n;

cout<<"Введите количество элементов: ";

cin>>n;

TMap m=make\_map(n);

print\_map(m);

cout<<"----------------------------------------------------------------------------\n";

//вычисление среднего

Time el=srednee(m);

cout<<"Среднее арифметичкеское: "<<el<<endl;

cout<<"----------------------------------------------------------------------------\n";

//добавлениевконец

m.insert(make\_pair(n,el));

print\_map(m);

int max=Max(m);

cout<<"----------------------------------------------------------------------------\n";

//удаление максимального

cout<<"Максимальный элемент: "<<m[max]<<" Его индекс: "<<max<<endl;

m.erase(max);

print\_map(m);

cout<<"----------------------------------------------------------------------------\n";

//Деление на минимальный

int min=Min(m);

cout<<"Минимальный элемент: "<<m[min]<<" Его индекс: "<<min<<endl;

delenie(m);

print\_map(m);

}

**Time.h**

#include<iostream>

usingnamespace std;

class Time

{

int min, sec;

public:

Time()

{

min=0;sec=0;

};

Time(int m, int s)

{

min=m;sec=s;

}

Time(const Time&t)

{

min=t.min;sec=t.sec;

}

~Time(){};

int get\_min()

{

return min;

}

int get\_sec()

{

return sec;

}

void set\_min(int m)

{

min=m;

}

void set\_sec(int s)

{

sec=s;

}

//перегруженныеоперации

Time&operator=(const Time&);

Time&operator++();

Time operator++(int);

Time operator+(const Time&);

Time operator/(constint);

Time operator/(const Time&);

booloperator<(const Time&);

booloperator>(const Time&);

friend istream&operator>>(istream&in, Time&t);

friend ostream&operator<<(ostream&out, const Time&t);

};

**Time.cpp**

#include"Time.h"

#include<iostream>

usingnamespace std;

//перегрузкаоперацииприсваивания

Time&Time::operator=(const Time&t)

{

//проверка на самоприсваивание

if(&t==this) return \*this;

min=t.min;

sec=t.sec;

return \*this;

}

//перегрузка префиксной операции инкремент

Time&Time::operator++()

{

int temp=min\*60+sec;

temp++;

min=temp/60;

sec=temp%60;

return \*this;

}

//перегрузка постфиксной операции инкремент

Time Time::operator ++(int)

{

int temp=min\*60+sec;

temp++;

Time t(min,sec);

min=temp/60;

sec=temp%60;

return t;

}

//перегрузкабинарнойоперациисложения

Time Time::operator+(const Time&t)

{

int temp1=min\*60+sec;

int temp2=t.min\*60+t.sec;

Time p;

p.min=(temp1+temp2)/60;

p.sec=(temp1+temp2)%60;

return p;

}

//перегрузкабинарнойоперацииделения

Time Time::operator/(constint t)

{

int temps=sec;

int tempm=min;

Time p;

p.min= tempm/t;

p.sec= temps/t;

return p;

}

//перегрузка бинарной операции деления

Time Time::operator/(const Time&t)

{

int temps=sec;

int tempm=min;

Time p;

p.min= tempm/t.min;

p.sec= tempm/t.sec;

return p;

}

//перегрузка операции "Меньше"

bool Time::operator<(const Time&t)

{

if (min==t.min)

{

if (sec<t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min<t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаоперации "Больше"

bool Time::operator>(const Time&t)

{

if (min==t.min)

{

if (sec>t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min>t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаглобальнойфункции-операцииввода

istream&operator>>(istream&in, Time&t)

{

cout<<"Минуты: ";

in>>t.min;

cout<<"Секунды: ";

in>>t.sec;

returnin;

}

//перегрузка глобальной функции-операции вывода

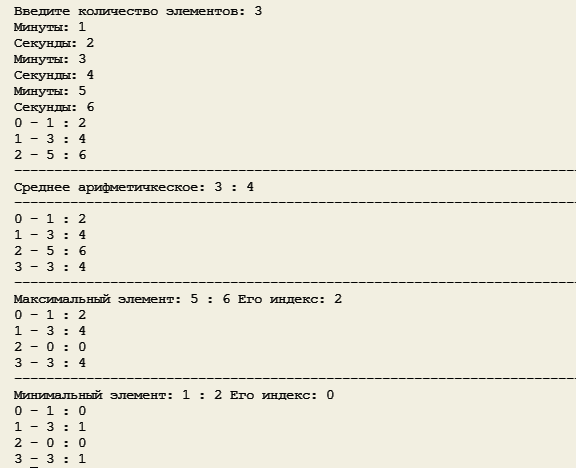
ostream&operator<<(ostream&out, const Time&t)

{

return (out<<t.min<<" : "<<t.sec);

}

**Скриншоты результатов работы программы**

****

## 18.12.3

**Постановка задачи**

(Вариант 10)

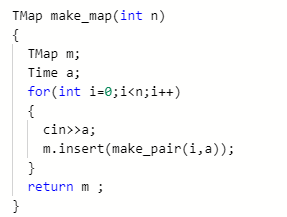
Задача 3

1. Контейнер -словарь.
2. ТипэлементовTime(см.лабораторнуюработу№3).
3. Найтисреднее арифметическоесловаряидобавить еговсловарьподномеромk.
4. Удалитьмаксимальныйэлементизсловаря.
5. Каждыйэлементразделитьнаминимальноезначениесловаря.
6. Выполнениевсехзаданийоформитьввидеметодовпараметризированногокласса.

**Анализ задачи**

**1.** Для решения задачи необходимо:

1.1. Создать словарь и заполнить его элеентами типаTime.



1.2.Реализовать функции для выполнения задания.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов словаря и их индексов использоваля тип Time.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Список элементов был представлен в виде словаря.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, выполняющий поставленные задачи, применяя стандартные алгоритмы и реализованные функции.



**Решение**

**Main.cpp**

#include"Time.h"

#include"Container.h"

#include<iostream>

usingnamespace std;

int main()

{

int n; //количество элементов в контейнере

cout<<"Введите количество элементов: ";

cin>>n;

Container <Time> v(n);//создать контейнер

v.Print();//напечатать контейнер

cout<<"----------------------------------------------------------------------------\n";

Time t=v.Srednee();//найти среднее арифметическое

cout<<"Среднее арифметическое: "<<t<<endl;

cout<<"Добавление среднего арифметического в массив"<<endl;

cout<<"Добавить на позицию: ";

int pos;

cin>>pos;//позиция для добавления

v.Add(pos,t);//добавление

v.Print();//печать

cout<<"----------------------------------------------------------------------------\n";

cout<<"Удаление максимального: "<<endl;

v.Del();

v.Print();

cout<<"----------------------------------------------------------------------------\n";

cout<<"Деление всех элементов на минимальный элемент: "<<endl;

v.Delenie();

v.Print();

}

**Container.h**

#include<iostream>

#include<map>

usingnamespace std;

template<class T>

class Container

{

map<int, T> v;

int len;

public:

Container(void);

Container(int n);

void Print();

~Container(void);

T Srednee();

void Add(int, T);

int Max();

void Del();

int Min();

void Delenie();

};

template<class T>

T Container<T>::Srednee()

{

Time s=v[0];

for(int i=1;i<v.size();i++) s=s+v[i];

int n=v.size();

return s/n;

}

template<class T>

void Container<T>::Add(int n, T el)

{

v.insert(make\_pair(n,el));

}

template<class T>

int Container<T>::Max()

{

typename map<int, T>::iterator i=v.begin();

int nom=0,k=0;

Time m=(\*i).second;

while(i!=v.end())

{

if(m<(\*i).second)

{

m=(\*i).second; nom=k;

}

i++;

k++;

}

return nom;

}

template<class T>

void Container<T>::Del()

{

int max=Max();

cout<<"max="<<v[max]<<" nom="<<max<<endl;

v.erase(max);

}

template<class T>

int Container<T>::Min()

{

typename map<int, T>::iterator i=v.begin();

int nom=0,k=0;

Time m=(\*i).second;

while(i!=v.end())

{

if((\*i).second.get\_min()!=0&&(\*i).second.get\_sec()!=0)

if(m>(\*i).second)

{

m=(\*i).second; nom=k;

}

i++;

k++;

}

return nom;

}

template<class T>

void Container<T>::Delenie()

{

T m=v[Min()];

cout<<"Min= "<<m<<endl;

for(int i=0;i<v.size();i++) v[i]=v[i]/m;

}

template<class T>

Container<T>::Container()

{

len=0;

}

template<class T>

Container<T>::~Container(void)

{

}

template<class T>

Container<T>::Container(int n)

{

T a;

for(int i=0;i<n;i++)

{

cin>>a;

v[i]=a;

}

len=v.size();

}

template<class T>

void Container<T>::Print()

{

for(int i=0;i<v.size();i++) cout<<i<<" - "<<v[i]<<" "<<endl;

cout<<endl;

}

**Time.h**

#include<iostream>

usingnamespace std;

class Time

{

int min, sec;

public:

Time()

{

min=0;sec=0;

};

Time(int m, int s)

{

min=m;sec=s;

}

Time(const Time&t)

{

min=t.min;sec=t.sec;

}

~Time(){};

int get\_min()

{

return min;

}

int get\_sec()

{

return sec;

}

void set\_min(int m)

{

min=m;

}

void set\_sec(int s)

{

sec=s;

}

//перегруженныеоперации

Time&operator=(const Time&);

Time&operator++();

Time operator++(int);

Time operator+(const Time&);

Time operator/(constint);

Time operator/(const Time&);

booloperator<(const Time&);

booloperator>(const Time&);

friend istream&operator>>(istream&in, Time&t);

friend ostream&operator<<(ostream&out, const Time&t);

};

**Time.cpp**

#include"Time.h"

#include<iostream>

usingnamespace std;

//перегрузкаоперацииприсваивания

Time&Time::operator=(const Time&t)

{

//проверка на самоприсваивание

if(&t==this) return \*this;

min=t.min;

sec=t.sec;

return \*this;

}

//перегрузка префиксной операции инкремент

Time&Time::operator++()

{

int temp=min\*60+sec;

temp++;

min=temp/60;

sec=temp%60;

return \*this;

}

//перегрузка постфиксной операции инкремент

Time Time::operator ++(int)

{

int temp=min\*60+sec;

temp++;

Time t(min,sec);

min=temp/60;

sec=temp%60;

return t;

}

//перегрузкаоперациисложения

Time Time::operator+(const Time&t)

{

int temp1=min\*60+sec;

int temp2=t.min\*60+t.sec;

Time p;

p.min=(temp1+temp2)/60;

p.sec=(temp1+temp2)%60;

return p;

}

Time Time::operator/(constint t)

{

int temps=sec;

int tempm=min;

Time p;

p.min= tempm/t;

p.sec= temps/t;

return p;

}

Time Time::operator/(const Time&t)

{

int temps=sec;

int tempm=min;

Time p;

p.min= tempm/t.min;

p.sec= tempm/t.sec;

return p;

}

bool Time::operator<(const Time&t)

{

if (min==t.min)

{

if (sec<t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min<t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

bool Time::operator>(const Time&t)

{

if (min==t.min)

{

if (sec>t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min>t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаглобальнойфункции-операцииввода

istream&operator>>(istream&in, Time&t)

{

cout<<"Минуты: ";

in>>t.min;

cout<<"Секунды: ";

in>>t.sec;

returnin;

}

//перегрузка глобальной функции-операции вывода

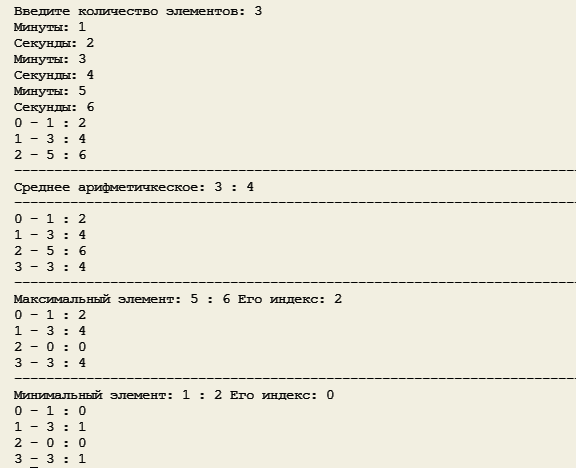
ostream&operator<<(ostream&out, const Time&t)

{

return (out<<t.min<<" : "<<t.sec);

}

**Скриншоты результатов работы программы**

****

**Ответы на контрольные вопросы**

**Что представляет собой ассоциативный контейнер?**

Ассоциативный массив содержит пары значений. Зная одно значение, называемое ключом, мы можем получить доступ к другому, называемому отображенным значением.

**Перечислить ассоциативные контейнеры библиотеки STL.**

Map, multimap (несколько элементов для одного ключа), set (множество, отсортирован), multiset (несколько элементов для одного ключа)

**Каким образом можно получить доступ к элементам ассоциативного контейнера?**

Ассоциативные контейнеры предоставляют стандартный доступ с помощью итераторов – прямых и обратных.

**Привести примеры методов, используемых в ассоциативных контейнерах.**

|  |  |
| --- | --- |
| **Операция или метод** | **Пояснение** |
| bool empty() const size\_type size() const  size\_type max\_size() | Методы определения размеров |
| insert() | Добавляет один элемент или диапазон элементов |
| erase() | Удаляет один элемент или диапазон элементов |
| clear() | Удаляет все элементы |
| swap() | Обмен данными с контейнером того же типа |
| key\_comp() value\_comp() | Возвращают объекты-функторы для сравнения  ключей и значений |
| find() count() lower\_bound()  upper\_bound() | Методы поиска (метод count() вычисляет для мульти контейнеров количество элементов с заданным ключом) |

**Каким образом можно создать контейнер map? Привести примеры.**

map<string, float> m1,

можно через

typedef map<string, float> myMap

myMap m2

**Каким образом упорядочены элементы в контейнере map по умолчанию? Как изменить порядок на обратный?**

Ассоциативный контейнер map требует, чтобы для типов ключа существовала операция “<” и он хранит свои элементы отсортированными по ключу (по убыванию).

Изменить порядок можно так:

map< string, int, greater<string> > myMap;

(по умолчанию тут lesser)

**Какие операции определены для контейнера map?**

Определены следующие операции: =, ==, <, <=, !=, >, >=.

**Написать функцию для добавления элементов в контейнер map с помощью функции make\_pair().**

TMap make\_map(int n)

{

TMap m;

int a;

for(int i=0;i<n;i++)

{

cout<<i<<"й элемент: ";

cin>>a;

m.insert(make\_pair(i,a));

}

return m;

}

**Написать функцию для добавления элементов в контейнер map с помощью функции операции прямого доступа [].**

#include <iostream>

#include <map>

using namespace std;

int main()

{

map <int,int> myMap;

for (int i=0; i<10; i++)

{

myMap[i]=10-i;

}

for (int i=0; i<10; i++)

{

cout<<myMap[i]<<' ';

}

}

**Написать функцию для печати контейнера map с помощью итератора.**

#include <iostream>

#include <map>

#include <iterator>

using namespace std;

int main()

{

map <int,int> myMap;

map <int,int>::iterator myIt;

for (int i=0; i<10; i++)

{

myMap[i]=10-i;

}

for (myIt=myMap.begin(); myIt!=myMap.end(); myIt++)

{

cout<<"Ключ= "<<(\*myIt).first<<" Значение= "<<(\*myIt).second<<'\n';

}

}

**Написать функцию для печати контейнера map с помощью функции операции прямого доступа [].**

#include <iostream>

#include <map>

using namespace std;

int main()

{

map <int,int> myMap;

for (int i=0; i<10; i++)

{

myMap[i]=10-i;

}

for (int i=0; i<10; i++)

{

cout<<myMap[i]<<' ';

}

}

**Чем отличаются контейнеры map и multimap?**

В multimap одному индексу может соответсвовать несколько значений.

**Что представляет собой контейнер set?**

В множестве хранятся объекты, упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта.

**Чем отличаются контейнеры map и set?**

Для контейнера set индексы – аттрибут элемента, для map индекс – первый член пары.

**Каким образом можно создать контейнер set? Привести примеры.**

set<int> set1; // создается пустое

множество int а[5] = { 1. 2. 3. 4, 5};

set<int> set2(a, а + 5);// инициализация копированием set<int>

set3(set2); // инициализация другим множеством

**Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?**

По умолчанию – по возрастанию.

Чтобы изменить порядок на обратный- set<int, greater<int>>

**Какие операции определены для контейнера set?**

**Написать функцию для добавления элементов в контейнер set.**

#include <iostream>

#include <set>

using namespace std;

typedef set<int>tset;

tset ::iterator i;

void main()

{

int a[4]={1,3,5,7};

tset s(a,a+4);

s.insert(10);

s.insert(6);

for(i=s.begin();i!=s.end();i++) cout<<\*i<<” ”; cout<<endl;

}

**Написать функцию для печати контейнера set.**

#include <iostream>

#include <set>

using namespace std;

typedef set<int>tset;

tset ::iterator i;

void main()

{

int a[4]={1,3,5,7};

tset s(a,a+4);

s.insert(10);

s.insert(6);

for(i=s.begin();i!=s.end();i++) cout<<\*i<<” ”; cout<<endl;

}

**Чем отличаются контейнеры set и multiset?**

Вотличии от set, в multiset ключи могут повторяться. Элементы с одинаковыми ключами хранятся в множестве в порядке их занесения.

## 18.13.1

**Постановка задачи**

(Вариант 10)

Задача 1.

1.  Создать последовательный контейнер.

2.  Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3.  Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4.  Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5.  Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

6.  Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).

7.  Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

8.    Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 1.

Контейнер – вектор (vector);

тип элементов - Time ;

Задание 3

Найти минимальный элемент и добавить его на

заданную позицию контейнера

Задание 4

Найти элементы большие среднего

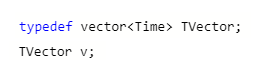
арифметического и удалить их из контейнера

Задание 5

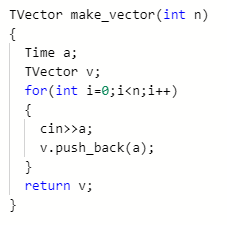
Каждый элемент домножить на максимальный элемент контейнера

**Анализ задачи**

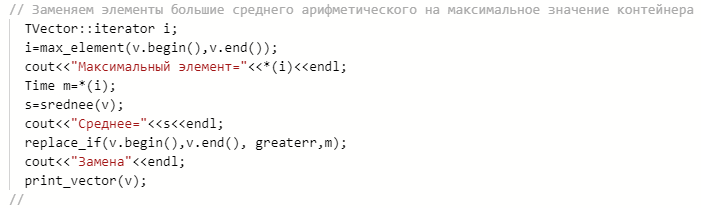
**1.** Для решения задачи необходимо:

1.1.  Создать последовательный контейнер.  


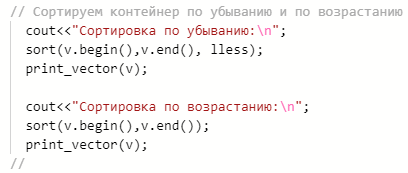
1.2.  Заполнить его элементами типа Time.



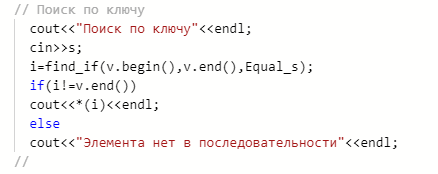
1.3. Заменить элементы большие среднего арифметического на максимальное значение контейнера



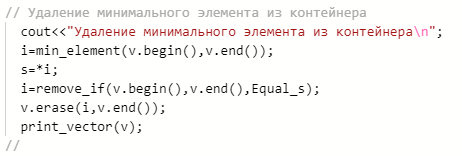
1.4. Отсортировать контейнер по убыванию и по возрастанию ключевого поля.



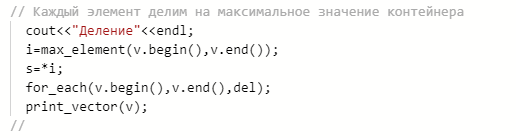
1.5. Найти в контейнере элемент с заданным ключевым полем.



1.6. Удалить минимальный элемент из контейнера.

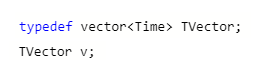


1.7. Каждый элемент разделить на максимальное значение контейнера.



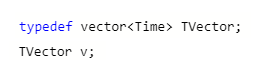
**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался пользовательский класс Time.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Набор элементов был представлен в виде вектора.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, осуществляющий применение функций и стандартных алгоритмов STL для выполнения поставленных задач.



**Решение**

**Main.cpp**

#include"Time.h"

#include<vector>

#include<iostream>

#include<algorithm>

usingnamespace std;

typedef vector<Time> TVector;

Time s;

bool greaterr(Time);

bool lless(Time,Time);

bool Equal\_s(Time);

TVector make\_vector(int);

void print\_vector(TVector);

Time srednee(TVector);

void del(Time&);

int main()

{

// Создание вектора

int n;

cout<<"Введите количество элементов: ";

cin>>n;

TVector v;

v=make\_vector(n);

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Заменяем элементы большие среднего арифметического на максимальное значение контейнера

TVector::iterator i;

i=max\_element(v.begin(),v.end());

cout<<"Максимальный элемент="<<\*(i)<<endl;

Time m=\*(i);

s=srednee(v);

cout<<"Среднее="<<s<<endl;

replace\_if(v.begin(),v.end(), greaterr,m);

cout<<"Замена"<<endl;

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Cортируем контейнер по убыванию и по возрастанию

cout<<"Сортировка по убыванию:\n";

sort(v.begin(),v.end(), lless);

print\_vector(v);

cout<<"Сортировка по возрастанию:\n";

sort(v.begin(),v.end());

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Поиск по ключу

cout<<"Поиск по ключу"<<endl;

cin>>s;

i=find\_if(v.begin(),v.end(),Equal\_s);

if(i!=v.end())

cout<<\*(i)<<endl;

else

cout<<"Элемента нет в последовательности"<<endl;

//

cout<<"----------------------------------------------------------------------------\n";

// Удаление минимального элемента из контейнера

cout<<"Удаление минимального элемента из контейнера\n";

i=min\_element(v.begin(),v.end());

s=\*i;

i=remove\_if(v.begin(),v.end(),Equal\_s);

v.erase(i,v.end());

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Каждый элемент делим на максимальное значение контейнера

cout<<"Деление"<<endl;

i=max\_element(v.begin(),v.end());

s=\*i;

for\_each(v.begin(),v.end(),del);

print\_vector(v);

//

}

TVector make\_vector(int n)

{

Time a;

TVector v;

for(int i=0;i<n;i++)

{

cin>>a;

v.push\_back(a);

}

return v;

}

void print\_vector(TVector v)

{

for(int i=0;i<v.size();i++) cout<<v[i]<<endl;

cout<<endl;

}

Time srednee(TVector v)

{

Time s=v[0];

for(int i=1;i<v.size();i++) s=s+v[i];

int n=v.size();

return s/n;

}

void del(Time& t)

{

t=t/s;

}

bool greaterr(Time t)

{

if (t>s) returntrue;

elsereturnfalse;

}

bool lless(Time t1,Time t2)

{

if (t1>t2) returntrue;

elsereturnfalse;

}

bool Equal\_s(Time t)

{

return t==s;

}

**Time.h**

#include<iostream>

usingnamespace std;

class Time

{

int min, sec;

public:

Time()

{

min=0;sec=0;

};

Time(int m, int s)

{

min=m;sec=s;

}

Time(const Time&t)

{

min=t.min;sec=t.sec;

}

~Time(){};

int get\_min()

{

return min;

}

int get\_sec()

{

return sec;

}

void set\_min(int m)

{

min=m;

}

void set\_sec(int s)

{

sec=s;

}

//перегруженныеоперации

Time&operator=(const Time&);

Time&operator++();

Time operator++(int);

Time operator+(const Time&);

Time operator/(constint);

Time operator/(const Time&);

booloperator<(const Time&);

booloperator>(const Time&);

booloperator==(const Time&);

friend istream&operator>>(istream&in, Time&t);

friend ostream&operator<<(ostream&out, const Time&t);

};

**Time.cpp**

#include"Time.h"

#include<iostream>

usingnamespace std;

//перегрузка операции присваивания

Time&Time::operator=(const Time&t)

{

min=t.min;

sec=t.sec;

return \*this;

}

//перегрузка операции инкремент (префикс)

Time&Time::operator++()

{

int temp=min\*60+sec;

temp++;

min=temp/60;

sec=temp%60;

return \*this;

}

//перегрузка операции инкремент (постфикс)

Time Time::operator ++(int)

{

int temp=min\*60+sec;

temp++;

Time t(min,sec);

min=temp/60;

sec=temp%60;

return t;

}

//перегрузкаоперациисложения

Time Time::operator+(const Time&t)

{

int temp1=min\*60+sec;

int temp2=t.min\*60+t.sec;

Time p;

p.min=(temp1+temp2)/60;

p.sec=(temp1+temp2)%60;

return p;

}

//перегрузкаоперацииделенияна int

Time Time::operator/(constint t)

{

int temps=sec;

int tempm=min;

Time p;

if (t!=0) p.min= tempm/t;

if (t!=0) p.sec= temps/t;

return p;

}

//перегрузка операции деления на Time

Time Time::operator/(const Time&t)

{

int temps=sec;

int tempm=min;

Time p;

if (t.min!=0) p.min= tempm/t.min;

else p.min=tempm;

if (t.sec!=0) p.sec= temps/t.sec;

else p.sec=temps;

return p;

}

//перегрузкаоперациименьше

bool Time::operator<(const Time&t)

{

if (min==t.min)

{

if (sec<t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min<t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаоперациибольше

bool Time::operator>(const Time&t)

{

if (min==t.min)

{

if (sec>t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min>t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаоперациисравнения

bool Time::operator==(const Time&t)

{

if (t.sec==sec && t.min==min)

{

returntrue;

}

else

{

returnfalse;

}

}

//перегрузкаглобальнойфункции-операцииввода

istream&operator>>(istream&in, Time&t)

{

cout<<"Минуты: ";

in>>t.min;

cout<<"Секунды: ";

in>>t.sec;

returnin;

}

//перегрузка глобальной функции-операции вывода

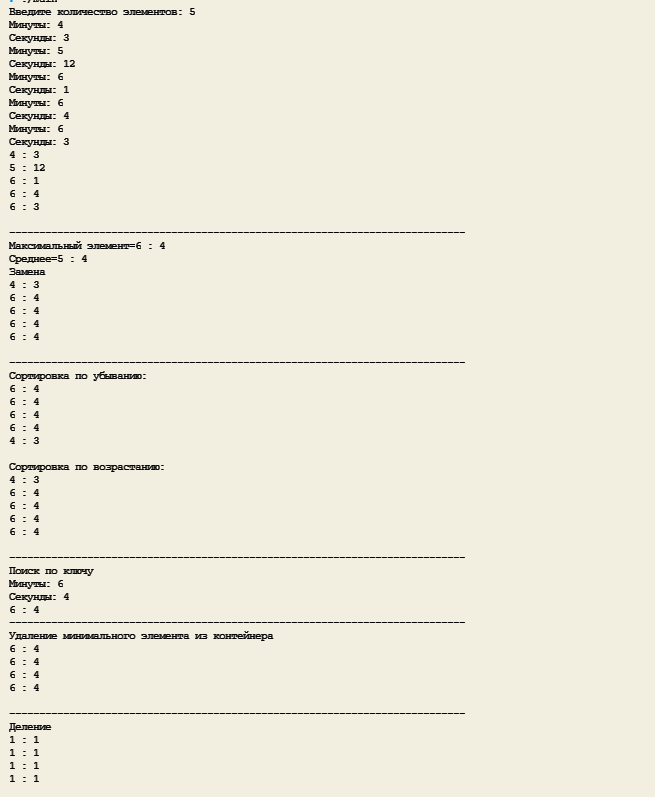
ostream&operator<<(ostream&out, const Time&t)

{

return (out<<t.min<<" : "<<t.sec);

}

**Скриншоты результатов работы программы**

****

## 18.13.2

**Постановка задачи**

(Вариант 10)

Задача 2.

1.  Создать адаптер контейнера.

2.  Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3.  Заменить элементы в соответствии с заданием (использовать алгоритмы

replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

4.  Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

5.  Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

6.  Найти в контейнере элемент с заданным ключевым полем (использоватьалгоритмы find(), find\_if(), count(), count\_if()).

7.  Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

8.    Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 2

Адаптер контейнера – стек

3.  Заменить элементы большие среднего арифметического на максимальное значение контейнера.

4.  Отсортировать контейнер по убыванию и по возрастанию ключевого поля.

5.  Найти в контейнере элемент с заданным ключевым полем.

6.  Удалить минимальный элемент из контейнера.

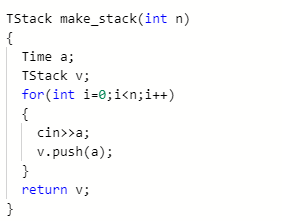
7.  Каждый элемент разделить на максимальное значение контейнера.

**Анализ задачи**

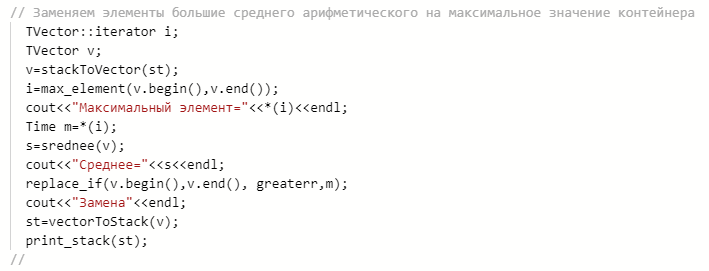
**1.** Для решения задачи необходимо:

1.1.  Создать адаптер контейнера.  

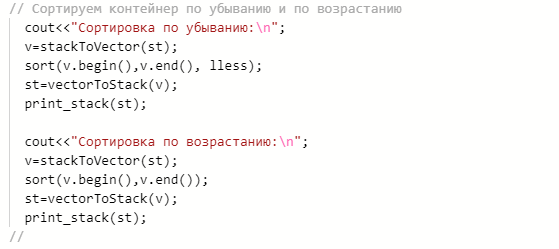

1.2.  Заполнить его элементами типа Time.



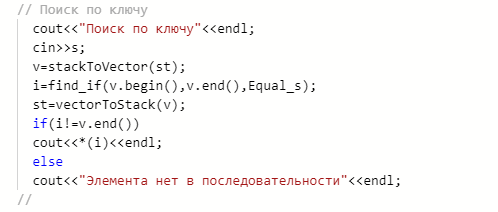
1.3. Заменить элементы большие среднего арифметического на максимальное значение контейнера



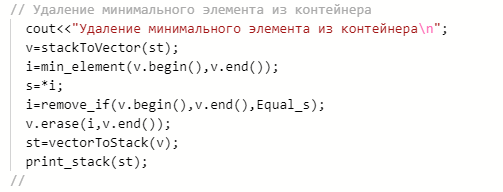
1.4. Отсортировать контейнер по убыванию и по возрастанию ключевого поля.



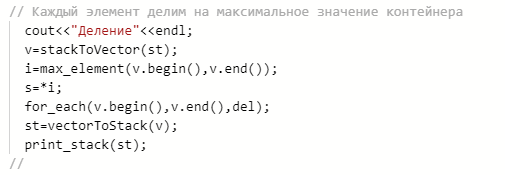
1.5. Найти в контейнере элемент с заданным ключевым полем.



1.6. Удалить минимальный элемент из контейнера.



1.7. Каждый элемент разделить на максимальное значение контейнера.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался пользовательский класс Time.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Набор элементов был представлен в виде стэка.



**4.**Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, осуществляющий применение функций и стандартных алгоритмов STL для выполнения поставленных задач.



**Решение**

**Main.cpp**

#include"Time.h"

#include<iostream>

#include<algorithm>

#include<stack>

#include<vector>

usingnamespace std;

typedef stack<Time> TStack;

typedef vector<Time> TVector;

Time s;

bool greaterr(Time);

bool lless(Time,Time);

bool Equal\_s(Time);

TStack make\_stack(int);

void print\_stack(TStack);

Time srednee(TVector);

void del(Time&);

TVector stackToVector(TStack);

TStack vectorToStack(TVector);

int main()

{

// Создание стека

int n;

cout<<"Введите количество элементов: ";

cin>>n;

TStack st;

st=make\_stack(n);

print\_stack(st);

//

cout<<"----------------------------------------------------------------------------\n";

// Заменяем элементы большие среднего арифметического на максимальное значение контейнера

TVector::iterator i;

TVector v;

v=stackToVector(st);

i=max\_element(v.begin(),v.end());

cout<<"Максимальный элемент="<<\*(i)<<endl;

Time m=\*(i);

s=srednee(v);

cout<<"Среднее="<<s<<endl;

replace\_if(v.begin(),v.end(), greaterr,m);

cout<<"Замена"<<endl;

st=vectorToStack(v);

print\_stack(st);

//

cout<<"----------------------------------------------------------------------------\n";

// Cортируем контейнер по убыванию и по возрастанию

cout<<"Сортировка по убыванию:\n";

v=stackToVector(st);

sort(v.begin(),v.end(), lless);

st=vectorToStack(v);

print\_stack(st);

cout<<"Сортировка по возрастанию:\n";

v=stackToVector(st);

sort(v.begin(),v.end());

st=vectorToStack(v);

print\_stack(st);

//

cout<<"----------------------------------------------------------------------------\n";

// Поиск по ключу

cout<<"Поиск по ключу"<<endl;

cin>>s;

v=stackToVector(st);

i=find\_if(v.begin(),v.end(),Equal\_s);

st=vectorToStack(v);

if(i!=v.end())

cout<<\*(i)<<endl;

else

cout<<"Элемента нет в последовательности"<<endl;

//

cout<<"----------------------------------------------------------------------------\n";

// Удаление минимального элемента из контейнера

cout<<"Удаление минимального элемента из контейнера\n";

v=stackToVector(st);

i=min\_element(v.begin(),v.end());

s=\*i;

i=remove\_if(v.begin(),v.end(),Equal\_s);

v.erase(i,v.end());

st=vectorToStack(v);

print\_stack(st);

//

cout<<"----------------------------------------------------------------------------\n";

// Каждый элемент делим на максимальное значение контейнера

cout<<"Деление"<<endl;

v=stackToVector(st);

i=max\_element(v.begin(),v.end());

s=\*i;

for\_each(v.begin(),v.end(),del);

st=vectorToStack(v);

print\_stack(st);

//

}

TStack make\_stack(int n)

{

Time a;

TStack v;

for(int i=0;i<n;i++)

{

cin>>a;

v.push(a);

}

return v;

}

void print\_stack(TStack v)

{

int n=v.size();

for(int i=0;i<n;i++)

{

cout<<v.top()<<endl;

v.pop();

}

cout<<endl;

}

Time srednee(TVector v)

{

Time s=v[0];

for(int i=1;i<v.size();i++) s=s+v[i];

int n=v.size();

return s/n;

}

void del(Time& t)

{

t=t/s;

}

bool greaterr(Time t)

{

if (t>s) returntrue;

elsereturnfalse;

}

bool lless(Time t1,Time t2)

{

if (t1>t2) returntrue;

elsereturnfalse;

}

bool Equal\_s(Time t)

{

return t==s;

}

TVector stackToVector(TStack st)

{

TVector v;

int i=0;

while (!st.empty())

{

v.push\_back(st.top());

st.pop();

i++;

}

return v;

}

TStack vectorToStack(TVector v)

{

TStack st;

int i=0;

while (!v.empty())

{

st.push(v[v.size()-1]);

v.erase(v.end());

}

return st;

}

**Time.h**

#include<iostream>

usingnamespace std;

class Time

{

int min, sec;

public:

Time()

{

min=0;sec=0;

};

Time(int m, int s)

{

min=m;sec=s;

}

Time(const Time&t)

{

min=t.min;sec=t.sec;

}

~Time(){};

int get\_min()

{

return min;

}

int get\_sec()

{

return sec;

}

void set\_min(int m)

{

min=m;

}

void set\_sec(int s)

{

sec=s;

}

//перегруженныеоперации

Time&operator=(const Time&);

Time&operator++();

Time operator++(int);

Time operator+(const Time&);

Time operator/(constint);

Time operator/(const Time&);

booloperator<(const Time&);

booloperator>(const Time&);

booloperator==(const Time&);

friend istream&operator>>(istream&in, Time&t);

friend ostream&operator<<(ostream&out, const Time&t);

};

**Time.cpp**

#include"Time.h"

#include<iostream>

usingnamespace std;

//перегрузкаоперацииприсваивания

Time&Time::operator=(const Time&t)

{

//проверка на самоприсваивание

if(&t==this) return \*this;

min=t.min;

sec=t.sec;

return \*this;

}

//перегрузка префиксной операции инкремент

Time&Time::operator++()

{

int temp=min\*60+sec;

temp++;

min=temp/60;

sec=temp%60;

return \*this;

}

//перегрузка постфиксной операции инкремент

Time Time::operator ++(int)

{

int temp=min\*60+sec;

temp++;

Time t(min,sec);

min=temp/60;

sec=temp%60;

return t;

}

//перегрузкабинарнойоперациисложения

Time Time::operator+(const Time&t)

{

int temp1=min\*60+sec;

int temp2=t.min\*60+t.sec;

Time p;

p.min=(temp1+temp2)/60;

p.sec=(temp1+temp2)%60;

return p;

}

Time Time::operator/(constint t)

{

int temps=sec;

int tempm=min;

Time p;

if (t!=0) p.min= tempm/t;

if (t!=0) p.sec= temps/t;

return p;

}

Time Time::operator/(const Time&t)

{

int temps=sec;

int tempm=min;

Time p;

if (t.min !=0)p.min= tempm/t.min;

else p.min=tempm;

if (t.sec !=0)p.sec= temps/t.sec;\

else p.sec=temps;

return p;

}

bool Time::operator<(const Time&t)

{

if (min==t.min)

{

if (sec<t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min<t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

bool Time::operator>(const Time&t)

{

if (min==t.min)

{

if (sec>t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min>t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

bool Time::operator==(const Time&t)

{

if (t.sec==sec && t.min==min)

{

returntrue;

}

else

{

returnfalse;

}

}

//перегрузкаглобальнойфункции-операцииввода

istream&operator>>(istream&in, Time&t)

{

cout<<"Минуты: ";

in>>t.min;

cout<<"Секунды: ";

in>>t.sec;

returnin;

}

//перегрузка глобальной функции-операции вывода

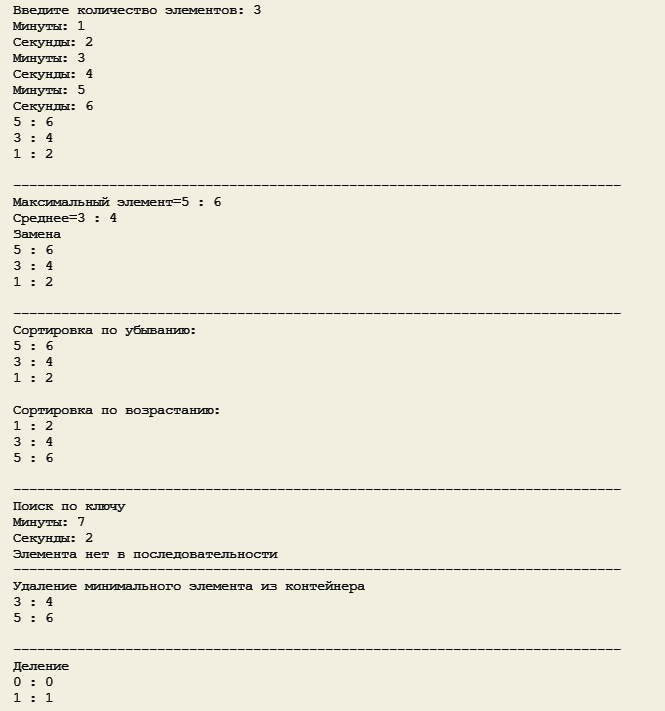
ostream&operator<<(ostream&out, const Time&t)

{

return (out<<t.min<<" : "<<t.sec);

}

**Скриншоты результатов работы программы**

****

## 18.13.3

**Постановка задачи**

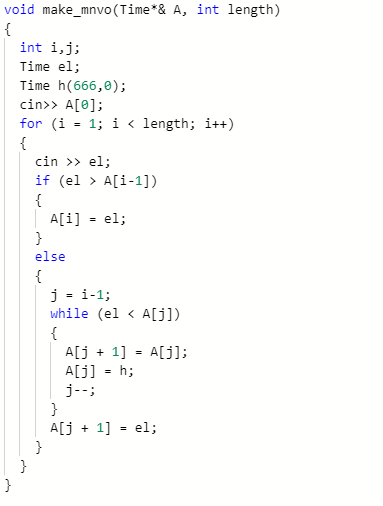
(Вариант 10)

**Анализ задачи**

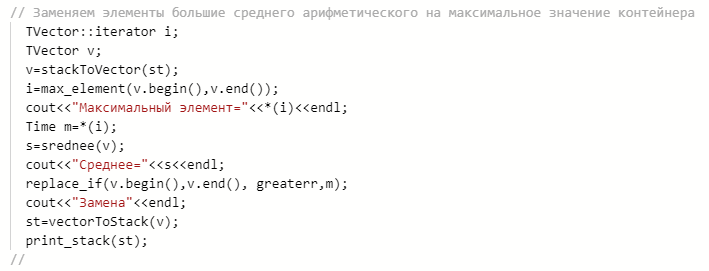
**1.** Для решения задачи необходимо:

1.1.  Создать ассоциативный контейнер.  

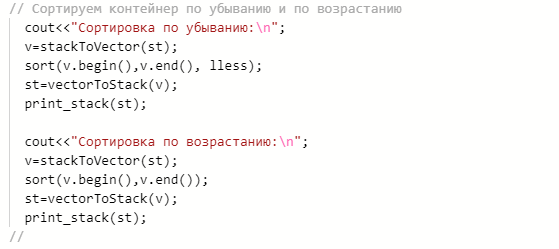

1.2.  Заполнить его элементами типа Time.



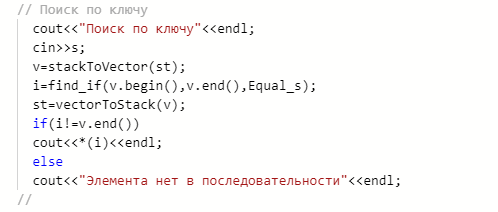
1.3. Заменить элементы большие среднего арифметического на максимальное значение контейнера



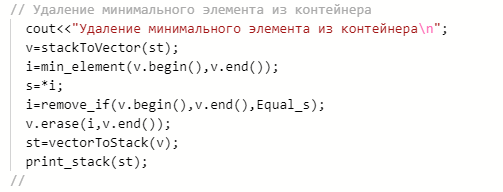
1.4. Отсортировать контейнер по убыванию и по возрастанию ключевого поля.



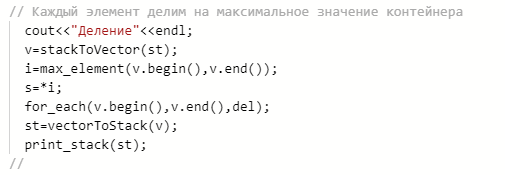
1.5. Найти в контейнере элемент с заданным ключевым полем.



1.6. Удалить минимальный элемент из контейнера.



1.7. Каждый элемент разделить на максимальное значение контейнера.



**2.**В ходе работы были использованы следующие типы данных:

2.1. Для хранения значений элементов использовался тип Time.



**3.**Для решения задачи данные были представлены в следующем виде:

3.1. Набор элементов был представлен в виде множества.



**4.** Для операций ввода и вывода использовались следующие операторы и функции:

4.1. Для операции ввода использовался оператор cin.



4.2. Для операции вывода использовался оператор cout.



**5.**Поставленные задачи будут решены следующими действиями:

5.1. Будет реализован код, осуществляющий применение функций и стандартных алгоритмов STL для выполнения поставленных задач.



**Решение**

**Main.cpp**

#include"Time.h"

#include"Mnvo.h"

#include<vector>

#include<iostream>

#include<algorithm>

usingnamespace std;

typedef vector<Time> TVector;

Time s;

bool greaterr(Time);

bool lless(Time,Time);

bool Equal\_s(Time);

TVector make\_vector();

void print\_vector(TVector);

Time srednee(TVector);

void del(Time&);

void make\_mnvo(Time\*&, int);

TVector mnvoToVector(Time\*&, int);

int main()

{

// Создание множества

int n;

cout<<"Введите количество элементов: ";

cin>>n;

Mnvo mn(3,1);

TVector v;

Time\*mas= new Time[n];

cout<<"Введите ваше множество\n";

make\_mnvo(mas,n);

cout<<'\n';

v=mnvoToVector(mas,n);

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Заменяем элементы большие среднего арифметического на максимальное значение контейнера

TVector::iterator i;

i=max\_element(v.begin(),v.end());

cout<<"Максимальный элемент="<<\*(i)<<endl;

Time m=\*(i);

s=srednee(v);

cout<<"Среднее="<<s<<endl;

replace\_if(v.begin(),v.end(), greaterr,m);

cout<<"Замена"<<endl;

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Cортируем контейнер по убыванию и по возрастанию

cout<<"Сортировка по убыванию:\n";

sort(v.begin(),v.end(), lless);

print\_vector(v);

cout<<"Сортировка по возрастанию:\n";

sort(v.begin(),v.end());

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Поиск по ключу

cout<<"Поиск по ключу"<<endl;

cin>>s;

i=find\_if(v.begin(),v.end(),Equal\_s);

if(i!=v.end())

cout<<\*(i)<<endl;

else

cout<<"Элемента нет в последовательности"<<endl;

//

cout<<"----------------------------------------------------------------------------\n";

// Удаление минимального элемента из контейнера

cout<<"Удаление минимального элемента из контейнера\n";

i=min\_element(v.begin(),v.end());

s=\*i;

i=remove\_if(v.begin(),v.end(),Equal\_s);

v.erase(i,v.end());

print\_vector(v);

//

cout<<"----------------------------------------------------------------------------\n";

// Каждый элемент делим на максимальное значение контейнера

cout<<"Деление"<<endl;

i=max\_element(v.begin(),v.end());

s=\*i;

for\_each(v.begin(),v.end(),del);

print\_vector(v);

//

}

TVector make\_vector(int n)

{

Time a;

TVector v;

for(int i=0;i<n;i++)

{

cin>>a;

v.push\_back(a);

}

return v;

}

void print\_vector(TVector v)

{

for(int i=0;i<v.size();i++) cout<<v[i]<<endl;

cout<<endl;

}

Time srednee(TVector v)

{

Time s=v[0];

for(int i=1;i<v.size();i++) s=s+v[i];

int n=v.size();

return s/n;

}

void del(Time& t)

{

t=t/s;

}

bool greaterr(Time t)

{

if (t>s) returntrue;

elsereturnfalse;

}

bool lless(Time t1,Time t2)

{

if (t1>t2) returntrue;

elsereturnfalse;

}

bool Equal\_s(Time t)

{

return t==s;

}

void make\_mnvo(Time\*& A, int length)

{

int i,j;

Time el;

Time h(666,0);

cin>> A[0];

for (i = 1; i < length; i++)

  {

    cin >> el;

    if (el > A[i-1])

    {

      A[i] = el;

    }

    else

    {

      j = i-1;

      while (el < A[j])

      {

        A[j + 1] = A[j];

        A[j] = h;

        j--;

     }

      A[j + 1] = el;

    }

  }

}

TVector mnvoToVector(Time\*& mas,int n)

{

TVector v;

for (int i=0; i<n; i++)

{

v.push\_back(mas[i]);

}

return v;

}

**Time.h**

#include<iostream>

usingnamespace std;

class Time

{

int min, sec;

public:

Time()

{

min=0;sec=0;

};

Time(int m, int s)

{

min=m;

if (s>59)sec=s%60;

min+=s/60;

}

Time(const Time&t)

{

min=t.min;sec=t.sec;

}

~Time(){};

int get\_min()

{

return min;

}

int get\_sec()

{

return sec;

}

void set\_min(int m)

{

min=m;

}

void set\_sec(int s)

{

sec=s;

}

//перегруженныеоперации

Time&operator=(const Time&);

Time&operator++();

Time operator++(int);

Time operator+(const Time&);

Time operator/(constint);

Time operator/(const Time&);

booloperator<(const Time&);

booloperator>(const Time&);

booloperator==(const Time&);

friend istream&operator>>(istream&in, Time&t);

friend ostream&operator<<(ostream&out, const Time&t);

};

**Time.cpp**

#include"Time.h"

#include<iostream>

usingnamespace std;

//перегрузка операции присваивания

Time&Time::operator=(const Time&t)

{

min=t.min;

sec=t.sec;

return \*this;

}

//перегрузка операции инкремент (префикс)

Time&Time::operator++()

{

int temp=min\*60+sec;

temp++;

min=temp/60;

sec=temp%60;

return \*this;

}

//перегрузка операции инкремент (постфикс)

Time Time::operator ++(int)

{

int temp=min\*60+sec;

temp++;

Time t(min,sec);

min=temp/60;

sec=temp%60;

return t;

}

//перегрузкаоперациисложения

Time Time::operator+(const Time&t)

{

int temp1=min\*60+sec;

int temp2=t.min\*60+t.sec;

Time p;

p.min=(temp1+temp2)/60;

p.sec=(temp1+temp2)%60;

return p;

}

//перегрузкаоперацииделенияна int

Time Time::operator/(constint t)

{

int temps=sec;

int tempm=min;

Time p;

if (t!=0) p.min= tempm/t;

if (t!=0) p.sec= temps/t;

return p;

}

//перегрузка операции деления на Time

Time Time::operator/(const Time&t)

{

int temps=sec;

int tempm=min;

Time p;

if (t.min!=0) p.min= tempm/t.min;

else p.min=tempm;

if (t.sec!=0) p.sec= temps/t.sec;

else p.sec=temps;

return p;

}

//перегрузкаоперациименьше

bool Time::operator<(const Time&t)

{

if (min==t.min)

{

if (sec<t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min<t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаоперациибольше

bool Time::operator>(const Time&t)

{

if (min==t.min)

{

if (sec>t.sec)

{

returntrue;

}

else

{

returnfalse;

}

}

else

{

if (min>t.min)

{

returntrue;

}

else

{

returnfalse;

}

}

}

//перегрузкаоперациисравнения

bool Time::operator==(const Time&t)

{

if (t.sec==sec && t.min==min)

{

returntrue;

}

else

{

returnfalse;

}

}

//перегрузкаглобальнойфункции-операцииввода

istream&operator>>(istream&in, Time&t)

{

cout<<"Минуты: ";

in>>t.min;

cout<<"Секунды: ";

in>>t.sec;

returnin;

}

//перегрузка глобальной функции-операции вывода

ostream&operator<<(ostream&out, const Time&t)

{

return (out<<t.min<<" : "<<t.sec);

}

**Mnvo.h**

#include<iostream>

usingnamespace std;

class Iterator

{

friendclass Mnvo;

public:

Iterator()

{

elem=0;

}

Iterator(const Iterator&it)

{

elem=it.elem;

}

booloperator==(const Iterator&it)

{

return elem==it.elem;

}

booloperator!=(const Iterator&it)

{

return elem!=it.elem;

};

voidoperator++()

{

++elem;

};

voidoperator--()

{

--elem;

}

int&operator \*() const

{

return\*elem;

}

int \*elem;

};

class Mnvo

{

public:

Mnvo (int length, int k=0);

Mnvo (Mnvo&);

~Mnvo ();

Mnvo&operator = (Mnvo&);

intoperator () ();

int&operator [] (int);

Mnvo operator - (Mnvo);

friend istream&operator>>(istream&,Mnvo&);

friend ostream&operator<<(ostream&,const Mnvo&);

Iterator first()

{

return beg;

}

Iterator last()

{

return end;

}

int size;

int\*data;

Iterator beg;

Iterator end;

};

**Mnco.cpp**

#include<iostream>

#include"Mnvo.h"

usingnamespace std;

Mnvo::Mnvo(int s,int k)

{

size=s;

data=newint[size];

for(int i=0;i<size;i++) data[i]=k;

beg.elem=&data[0];

end.elem=&data[s-1];

}

Mnvo::Mnvo(Mnvo& a)

{

size=a.size;

data=newint[size];

for(int i=0;i<size;i++) data[i]=a.data[i];

beg.elem=&data[0];

end.elem=&data[0];

}

Mnvo::~Mnvo()

{

delete[]data;

data=0;

}

Mnvo &Mnvo::operator=(Mnvo&a)

{

if(this==&a) return \*this;

size=a.size;

if (data!=0) delete[]data;

data=newint[size];

for(int i=0;i<size;i++) data[i]=a.data[i];

return \*this;

}

int& Mnvo::operator [] (int index)

{

if (index<size) return data[index];

else cout<<"\nIndex>size";

return data[0];

}

int Mnvo::operator ()()

{

return size;

}

Mnvo Mnvo::operator - (Mnvo a)

{

bool f;

Mnvo mas((\*this).size);

for (int i=0; i<a.size; i++)

{

f=false;

for (int j=0; j<(\*this).size; j++)

{

if (a[i]==(\*this)[j] && !f)

{

(\*this)[j]=-666;

f=true;

}

}

}

int l=0;

for (int k=0; k<(\*this).size; k++)

{

if ((\*this)[k]!=-666)

{

mas.data[l]=(\*this)[k];

l++;

}

}

return mas;

}

ostream&operator<<(ostream&out,const Mnvo&a)

{

for(int i=0;i<a.size;++i)

{

if (a.data[i]!=-666) out<<a.data[i]<<" ";

}

return out;

}

istream&operator>>(istream&in,Mnvo&a)

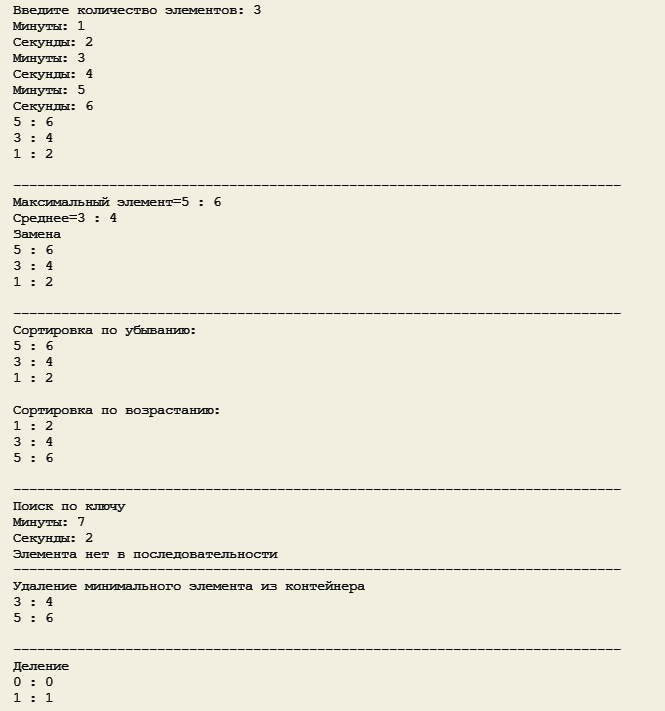
{

for(int i=0;i<a.size;++i) in>>a.data[i];

returnin;

}

**Скриншоты результатов работы программы**

****

**GitHub**

[**https://github.com/sugarrrfqs/Lab18**](https://github.com/sugarrrfqs/Lab18)