Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Лабораторная работа №7

Шаблоны классов

Выполнил

Студент группы РИС-22-2б

Вековшинин Д. А.

Проверила

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

1. Определить шаблон класса-контейнера (см. лабораторную работу №6).

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 (см. лабораторную работу №3).

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

Вариант 10: Класс- контейнер Вектор с элементами типа int.

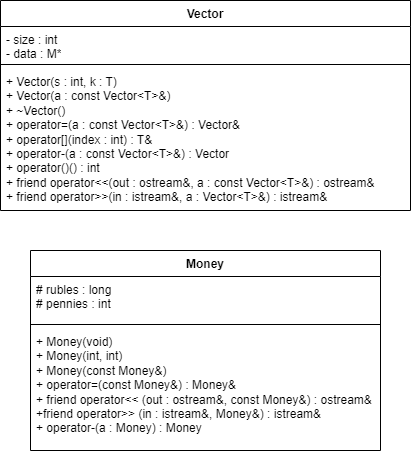
Реализовать операции:

[] – 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.

UML-диаграмма



Определение компонентных функций класса Vector

//конструктор с параметрами

template <class T>

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

{

size = s;

data = new T[size];

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

data[i] = k;

}

//конструктор копирования

template <class T>

Vector<T>::Vector(const Vector& a)

{

size = a.size;

data = new T[size];

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

data[i] = a.data[i];

}

//деструктор

template <class T>

Vector<T>::~Vector()

{

delete[]data;

data = 0;

}

//операция присваивания

template <class T>

Vector<T>& Vector<T>::operator=(const Vector<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;

}

//операция доступа по индексу

template <class T>

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

{

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

else cout << "\nError! Index>size";

}

//операция разности множеств

template <class T>

Vector<T> Vector<T>::operator-(const Vector<T>& a)

{

Vector<T> temp(size, \*data);

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

temp.data[i] = data[i] - a.data[i];

return temp;

}

//операция для получения длины вектора

template <class T>

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

{

return size;

}

//операции для ввода-вывода

template <class T>

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

{

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

out << a.data[i] << " ";

return out;

}

template <class T>

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

{

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

in >> a.data[i];

return in;

}

Определение компонентных функций класса Money

Money::Money(void)

{

rubles = pennies = 0;

}

Money::Money(int R, int P)

{

rubles = R;

pennies = P;

}

Money::Money(const Money& m)

{

rubles = m.rubles;

pennies = m.pennies;

}

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

{

rubles = m.rubles;

pennies = m.pennies;

return\*this;

}

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

{

out << m.rubles << "," << m.pennies;

return out;

}

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

{

cout << "\nrubles? "; in >> m.rubles;

cout << "\npennies? "; in >> m.pennies;

return in;

}

Money Money::operator-(Money a)

{

int first = rubles \* 100 + pennies;

int second = a.rubles \* 100 + a.pennies;

first -= second;

Money temp(first / 100, first % 100);

return temp;

}

Функция main()

void main()

{

cout << "Enter set of 5 values: " << endl;

//инициализация, ввод и вывод значений вектора

Vector<long>A(5, 0);

cin >> A;

cout << A << endl;

//инициализация и вывод значений вектора

Vector <long>B(10, 1);

cout << B << endl;

//операция присваивания

B = A;

cout << B << endl;

A = { 8, 10 };

//доступ по индексу

cout << A[2] << endl;

//получение длины вектора

cout << "size= " << A() << endl;

Vector<long>C(10, 0);

//операция разности множеств

C = B - A;

cout << C << endl;

Money m;

cin >> m;

cout << m;

Money first;

cout << "\nfirst: ";

cin >> first;

Money second;

cout << "\nsecond: ";

cin >> second;

Money third;

third = first - second;

cout << "third: " << third;

Vector<Money> D(5, m);

cin >> D;

cout << D << endl;

Vector<Money> E(10, m);

cout << E << endl;

E = D;

cout << E << endl;

cout << D[2] << endl;

cout << "size=" << D() << endl;

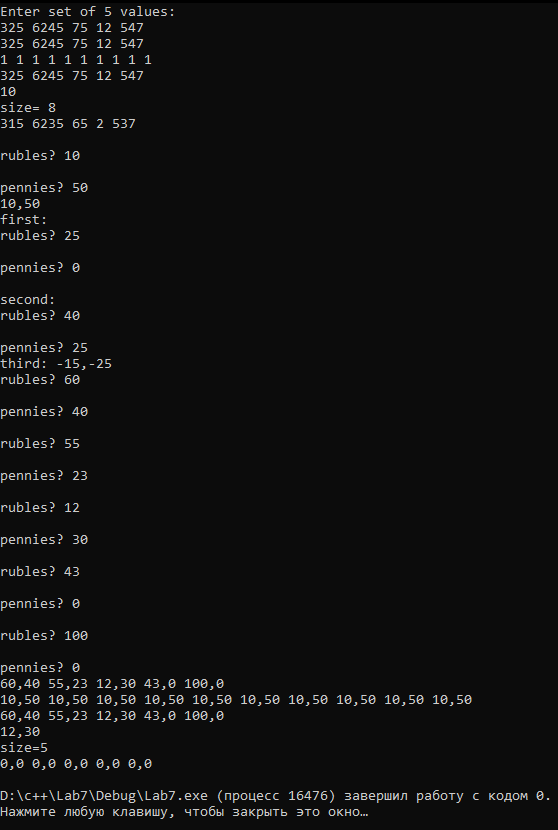
Vector<Money> F(10, m);

F = D - E;

cout << F << endl;

}

Результаты работы программы



Ответы на контрольные вопросы

1. В чем смысл использования шаблонов?

С помощью шаблона функций можно отделить алгоритм от конкретных типов данных, передавая тип в качестве параметра. Шаблоны классов предоставляют аналогичную возможность, позволяя создавать параметризированные классы.

1. Каковы синтаксис/семантика шаблонов функций?

Формат шаблона:

template <параметры\_шаблона>

заголовок\_функции

{тело функции}

Таким образом, шаблон семейства функций состоит из 2 частей – заголовка

шаблона: template<список параметров шаблона> и обыкновенного определения функции, в котором вместо типа возвращаемого значения и/или типа параметров, записывается имя типа, определенное в заголовке шаблона.

Пример:

шаблон функции, которая находит абсолютное значение числа любого типа

template<class type>//type – имя параметризируемого типа

type abs(type x)

{

if(x<0)return -x;

else return x;

}

1. Каковы синтаксис/семантика шаблонов классов?

Определение шаблонного (обобщенного) класса имеет вид:

template <параметры шаблона>

class имя\_класса

{…};

Пример:

template<classT>

class Point

{

T x,y;//координаты точки

public:

Point(T X=0,T Y=0):x(X),y(Y){}

void Show ();

};

template<classT>

void Point::Show()

{

cout<<”(“<<x<<” , ”<<y<<”)”;

}

1. Что такое параметры шаблона функции?

Параметризируемые типы, передаваемые шаблону, чтобы отделить алгоритм от конкретных типов данных

1. Перечислите основные свойства параметров шаблона функции

Шаблон функции определяется один раз, но определение параметризируется, т. е. тип данных передается как параметр шаблона.

Шаблон служит для автоматического формирования конкретных описаний

функций по тем вызовам, которые компилятор обнаруживает в программе.

1. Как записывать параметр шаблона?

Определение шаблонного (обобщенного) класса имеет вид:

template <параметры шаблона>

class имя\_класса

{…};

1. Можно ли перегружать параметризованные функции?

При перегрузке функции для каждого используемого типа определяется своя функция.

1. Перечислите основные свойства параметризованных классов

Механизм шаблонов в С++ допускает применение абстрактного

типа в качестве параметра при определении класса. После того как шаблон класса определен, он может использоваться для определения конкретных классов.

При включении шаблона класса в программу никакие классы на самом деле не генерируются до тех пор, пока не будет создан экземпляр шаблонного класса, в котором вместо параметра шаблона указывается конкретный тип. Экземпляр создается либо объявлением объекта, либо объявлением указателя на инстанцированный шаблонный тип с присваиванием ему адреса с помощью операции new

1. Все ли компонентные функции параметризованного класса являются параметризованными?

Только те, определение которых параметризируется  
Пример.

шаблон функции, которая находит абсолютное значение числа любого типа

template<class type>//type – имя параметризируемого типа

type abs(type x)

{

if(x<0)return -x;

else return x;

}

1. Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?

Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, т.е. по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону.

1. Могут ли шаблоны классов содержать виртуальные компонентные функции?

Шаблоны методов (функций) не могут быть виртуальными.

1. Как определяются компонентные функции параметризованных классов вне определения шаблона класса?

Реализация компонентной функции шаблона класса, которая находится вне определения шаблона класса, должна включать дополнительно следующие два элемента:

* Определение должно начинаться с ключевого слова template, за которым следует такой же список\_параметров\_типов в угловых скобках, какой указан в определении шаблона класса.
* За именем\_класса, предшествующим операции области видимости (::), должен следовать список\_имен\_параметров шаблона.

template<список\_типов>

тип\_возвр\_значения имя\_класса<список\_имен\_параметров>::имя\_функции(список\_параметров)

{ ... }

1. Что такое инстанцирование шаблона

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

1. На каком этапе происходит генерирование определения класса по шаблону?

При создании экземпляра шаблонного класса, в котором

вместо параметра шаблона указывается конкретный тип

Пример:

template<classT>

class Point

{

T x,y;//координаты точки

public:

Point(T X=0,T Y=0):x(X),y(Y){}

void Show ();

};

template<classT>

void Point::Show()

{

cout<<”(“<<x<<” , ”<<y<<”)”;

}

Point <int> a(13,15);

Point <float>\*pa=new Point<float>(10.1,0.55);

Встретив такие объявления, компилятор генерирует код исходного класса.