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

Тема: «Лабораторная работа №7»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

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 операции для контейнера, содержащего элементы стандартных типов данных.
6. Реализовать пользовательского класса операции ввода-вывода.
7. Перегрузить для пользовательского класса операции ввода-вывода.
8. Перегрузить операции необходимые для выполнения операции контейнерного класса.
9. Инстанцировать шаблон для пользовательского класса.
10. Написать тестирующую программу, иллюстрирующую выполнение операции для контейнера, содержащего элементы пользовательского класса.

**Вариант 8:**

Класс-контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

[] - доступа по индексу;

== - проверка на равенство;

> - число - принадлежность числа множеству.

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.

**Исходный код программы:**

**Файл “Set.h”:**

#include <iostream>

using namespace std;

template <class T>

class Set {

public:

Set(int s, T k);

Set();

~Set();

Set<T>& operator |=(T key);

Set& operator =(const Set<T>& other);

bool operator !=(const Set<T>& other);

bool operator ==(const Set<T>& other);

bool operator <(T key) const;

T& operator [](int index);

template <class T2>

friend ostream& operator << (ostream&, const Set<T2>& s);

private:

T\* elements;

int size;

};

**Файл “Set.cpp”:**

#include "Set.h"

template <class T>

Set<T>::Set(int s, T k) {

size = s;

elements = new T[size];

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

elements[i] = k;

}

template <class T>

Set<T>::Set() {

size = 0;

elements = new T[size];

}

template <class T>

Set<T>::~Set() {

delete[] elements;

size = 0;

}

template <class T>

Set<T>& Set<T>::operator =(const Set<T>& other) {

if (this == &other)

return \*this;

size = other.size;

if (elements != 0)

delete[] elements;

elements = new T[size];

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

elements[i] = other.elements[i];

return \*this;

}

template <class T>

bool Set<T>::operator !=(const Set<T>& other) {

bool flag;

if (size != other.size)

return false;

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

flag = (elements[i] == other.elements[i]);

return flag;

}

template <class T>

bool Set<T>::operator ==(const Set<T>& other) {

if (size != other.size)

return false;

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

if (elements[i] != other.elements[i])

return false;

return true;

}

template <class T>

bool Set<T>::operator <(T key) const {

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

if (key == elements[i])

return true;

return false;

}

template <class T>

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

if (index < size)

return elements[index];

else

throw out\_of\_range("Index doesn't exist");

}

template <class T>

Set<T>& Set<T>::operator |=(T key) {

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

if (key == elements[i])

return \*this;

int i, k;

for (k = 0; k < size && elements[k] <= key; k++);

if (k == size) {

elements[size] = key;

}

else {

for (i = size; i > k; i--)

elements[i] = elements[i - 1];

elements[k] = key;

}

size++;

return \*this;

}

template <class T>

ostream& operator <<(ostream& out, const Set<T>& s) {

out << "[ ";

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

out << s.elements[i] << " ";

out << "]\n";

return out;

}

**Файл “Money.h”:**

#include <iostream>

using namespace std;

class Money {

long int rub;

int kop;

public:

Money() {

rub = 0;

kop = 0;

}

Money(int rub, int kop) {

this->rub = rub;

this->kop = kop;

}

Money(const Money &other) {

rub = other.rub;

kop = other.kop;

}

~Money() {};

Money& operator=(const Money&);

Money operator+(const Money&);

Money operator-(const Money&);

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

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

};

**Файл “Money.cpp”:**

#include "Money.h"

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

if (&t == this)

return \*this;

rub = t.rub;

kop = t.kop;

return \*this;

}

Money Money::operator+(const Money&t) {

int temp1 = rub \* 100 + kop;

int temp2 = t.rub \* 100 + t.kop;

Money p;

p.rub = (temp1 + temp2) / 100;

p.kop = (temp1 + temp2) % 100;

return p;

}

Money Money::operator-(const Money&t) {

int temp1 = rub \* 100 + kop;

int temp2 = t.rub \* 100 + t.kop;

Money p;

p.rub = (temp1 - temp2) / 100;

p.kop = (temp1 - temp2) % 100;

return p;

}

istream&operator>>(istream&in, Money&t) {

cout << "Rubles: ";

in >> t.rub;

cout << "Kopecks: ";

in >> t.kop;

return in;

}

ostream&operator<<(ostream&out, const Money&t) {

return (out << t.rub << "." << t.kop << " rub.");

}

**Файл “Lab\_7.cpp”:**

#include "Set.cpp"

#include "Money.cpp"

#include <iostream>

using namespace std;

int main() {

Set<int> A;

(A |= 1) |= 10;

cout << "A: " << A << "\n";

Set<int> B;

((B |= 2) |= 1) |= 10;

cout << "B: " << B << "\n";

A |= 2;

if (A==B)

cout << "A == B\n";

else

cout << "A != B\n";

B[1] = 3;

cout << "B[1]: " << B[1] << "\n";

if (A==B)

cout << "A == B\n";

else

cout << "A != B\n";

if (A<2)

cout << "A < 2\n";

else

cout << "A != 2\n";

Money m1;

cin >> m1;

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

Money m2;

cin >> m2;

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

cout << m1 + m2 << "\n";

Set<Money> C(1, m1);

Set<Money> D(1, m2);

cout << "\nC: " << C << "\n";

cout << "D: " << D << "\n";

D = C;

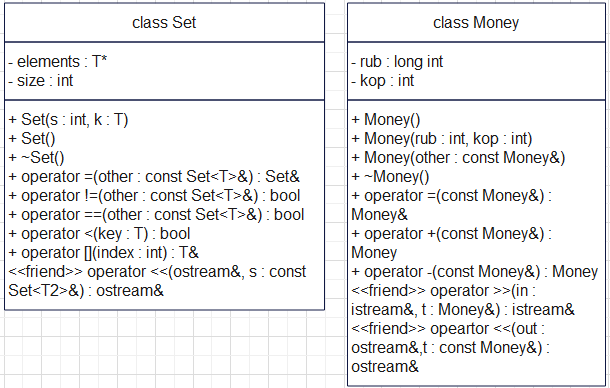
cout << "D: " << D << "\n";

cout << "C[0]: " << C[0] << "\n";

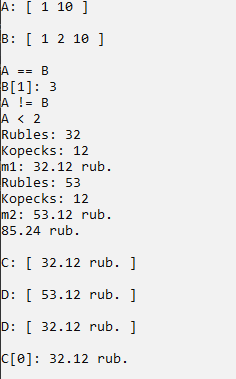
return 0;

}

**UML диаграмма:**



**Скриншот результата выполнения программы**



**Ответы на вопросы:**

1. В чем смысл использования шаблонов?

**Ответ:** Шаблоны вводятся для того, чтобы автоматизировать создание функций, обрабатывающих разнотипные данные.

1. Каковы синтаксис/семантика шаблонов функций?

**Ответ:**

template <параметры\_шаблона>

заголовок\_функции

{тело функции}

1. Каковы синтаксис/семантика шаблонов классов?

**Ответ:**

template <параметры\_шаблона>

class имя\_класса

{...};

1. Что такое параметры шаблона функций?

**Ответ:** Параметры шаблона могут быть типами данных или константами, которые используются внутри шаблонной функции.

1. Перечислите основные свойства параметров шаблона функции.

**Ответ:**

* В списке параметров шаблона может быть несколько параметров, каждому предшествует ключевое слово typename или class.
* Имя параметра шаблона имеет в определяемой шаблоном функции все права имени типа, то есть с его помощью могут специализироваться формальные примеры, определяться тип возвращаемого функцией значений и типы любых объектов, локализованных в теле функции.
* Параметризированная функция может иметь сколько угодно непараметризированных формальных параметров.
* В списке описания прототипа шаблона имена параметров не обязаны совпадать с именами в описании шаблона.
* При конкретизации шаблонного определения функции необходимо, чтобы при вызове функции типы фактических параметров, соответствующие одинаково параметризованным формальным параметрам, были одинаковы.

1. Как записывать параметр шаблона?

**Ответ:** В списке параметров шаблона может быть несколько параметров, каждому предшествует ключевое слово typename или class.

template <class T1, T2>

1. Можно ли перегружать параметризованные функции?

**Ответ:** При использовании шаблонов возможна перегрузка как шаблонов, так и функций.

1. Перечислите основные свойства параметризованных классов.

**Ответ:**

* Шаблон класса используется для построения родового (параметризованного) класса.
* При описании родового класса создаётся семейство родственных классов, которые применимы к любому типу данных.
* Механизм шаблонов позволяет использовать абстрактный тип данных в качестве параметра шаблона.
* В определении класса, входящего шаблон, имя класса является не именем отдельного класса, а параметризованным именем семейства классов.
* После того, как шаблон класса определён, он может использоваться для определения конкретных классов. Процесс генерации компилятором определения конкретного класса по шаблону класса и параметрам шаблона называется инстанцированием шаблона.

1. Все ли компонентные функции параметризованного класса являются параметризованными?

**Ответ:** Компонентные функции параметризованного класса автоматически являются параметризованными.

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

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

1. Могут ли шаблоны классов содержать виртуальные компонентные функции?

**Ответ:** Шаблоны функции нельзя описывать как виртуальные.

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

**Ответ:**

template <список\_параметров> тип\_функции имя\_класса <список\_имён\_параметров> :: имя\_функции (список параметров) {

тело функции

}

1. Что такое инстанцирование шаблона?

**Ответ:** Инстанцирование шаблона - процесс генерации компилятором определения конкретного класса по шаблону класса и параметрам шаблона.

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

**Ответ:**