Министерство науки и высшего образования РФ

Федеральное бюджетное государственное образовательное учреждение высшего профессионального образования

Пермский национальный исследовательский политехнический университет (ПНИПУ)

Факультет: Электротехнический (ЭТФ)

Направление: 09.03.01 «Информатика и вычислительная техника (Автоматизированные системы обработки информации и управления)»

Кафедра: «Информационные технологии и автоматизированных систем» (ИТАС)

Информатика

Лабораторная работа №7

Шаблоны классов

Студент: Балтаев Э. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Группа: АСУ-20-1бзу

Работу проверил:

доцент кафедры ИТАС

Полякова О. А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_

Пермь 2021г.

Постановка задачи

1. Определить шаблон класса-контейнер (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 класса

Задание варианта

Класс-контейнер МНОЖЕСТВО с элементами типа int

Реализовать операции:

[] – доступа по индексу

Int() – определение размера вектора;

\* - пересечение множеств

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

Описание класса

1) Сумма

#pragma once

#include <iostream>

using namespace std;

class money

{

private:

long rub;

int cop;

public:

money();

money(long r, int c);

money(const money& m);

~money() {}

long get\_rub() { return rub; }

long get\_cop() { return cop; }

void set\_rub(long r);

void set\_cop(int c);

money& operator= (const money& m);

bool operator== (const money& m);

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

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

};

2) Множество

#pragma once

#include <iostream>

using namespace std;

template <class T>

class Set

{

private:

int size;

T\* data;

public:

Set(int s);

Set(const Set<T>& s);

~Set();

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

friend ostream& operator<< <>(ostream& out, const Set<T>& v);

friend istream& operator>> <>(istream& in, Set<T>& v);

T& operator[] (int index); // доступ по индексу

int operator() (); // определение размера

Set<T> operator\* (const Set<T>& s); // пересечение множеств

private:

void reset(); // убрать одинаковые

};

Определение компонентных функций

1) Сумма

#include "money.h"

money::money()

{

rub = 0;

cop = 0;

}

money::money(long r, int c)

{

set\_rub(r);

set\_cop(c);

}

money::money(const money& m)

{

rub = m.rub;

cop = m.cop;

}

void money::set\_rub(long r)

{

if (r < 0)

rub = 0;

else

rub = r;

}

void money::set\_cop(int c)

{

if (c < 0)

cop = 0;

else {

rub += c / 100;

cop = c % 100;

}

}

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

{

if (&m == this)

return \*this;

rub = m.rub;

cop = m.cop;

return \*this;

}

bool money::operator==(const money& m)

{

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

return true;

return false;

}

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

{

long r;

int c;

cout << "Введите сумму через пробел: ";

in >> r; m.set\_rub(r);

in >> c; m.set\_cop(c);

return in;

}

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

{

out << m.rub << "," << m.cop;

return out;

}

2) Множество (в том же файле, что и определение)

template <class T>

Set<T>::Set(int s)

{

size = s;

data = new T[size];

}

template <class T>

Set<T>::Set(const Set<T>& s)

{

size = s.size;

data = new T[size];

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

data[i] = s.data[i];

}

template <class T>

Set<T>::~Set()

{

delete[] data;

data = 0;

}

template <class T>

Set<T>& Set<T>::operator=(const Set<T>& v)

{

if (this == &v)

return \*this;

if (size != 0)

delete[] data;

size = v.size;

data = new T[size];

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

data[i] = v.data[i];

return \*this;

}

template <class T>

void Set<T>::reset()

{

int ok, k = 0;

T\* arr = new T[size];

// идем по элементами

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

ok = 0;

// идем по доп. массиву

for (int j = 0; j < k; j++)

// если он там есть

if (data[i] == arr[j]) {

ok = 1; // отмечаем, что копировать не надо

break;

}

// если его там нет

if (ok == 0) {

arr[k] = data[i];

k++;

}

}

// копируем

if (size != 0)

delete[] data;

size = k;

data = new T[size];

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

data[i] = arr[i];

delete[] arr;

}

template <class T>

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

{

if (index < 0 || index >= size) {

cout << "Неверный индекс";

}

else

return data[index];

}

template <class T>

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

{

return size;

}

template <class T>

Set<T> Set<T>::operator\*(const Set<T>& s)

{

int count, k = 0;

if (size >= s.size)

count = size;

else

count = s.size;

T\* arr = new T[count];

// идем по элементами

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

// идем по элементами другого множества

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

// если они равны

if (data[i] == s.data[j]) {

arr[k] = data[i]; // копируем

k++;

break;

}

// копируем

Set temp(k);

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

temp.data[i] = arr[i];

delete[] arr;

return temp;

}

template <class T>

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

{

if (v.size == 0)

out << "Пусто";

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

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

return out;

}

template <class T>

istream& operator>>(istream& in, Set<T>& v)

{

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

in >> v.data[i];

}

v.reset();

return in;

}

Определение главного файла

#include <iostream>

#include "Set.h"

#include "Money.h"

int main()

{

system("chcp 1251");

cout << "\nСтандартные типы данных: int: 5, 3; double: 2; float: 2\n";

cout << "Введите элементы через пробел\n";

Set<int> sInt1(5);

cin >> sInt1;

Set<int> sInt2(3);

cin >> sInt2;

Set<double> sDouble(2);

cin >> sDouble;

Set<float> sFloat(2);

cin >> sFloat;

cout << "sInt1: " << sInt1 << endl;

cout << "sInt2: " << sInt2 << endl;

cout << "sDouble: " << sDouble << endl;

cout << "sFloat: " << sFloat << endl;

cout << "\nОперации над int\n";

cout << "sInt1 size = " << sInt1() << endl;

cout << "sInt[3] = " << sInt1[3] << endl;

cout << "sInt1 \* sInt2: " << (sInt1 \* sInt2) << endl;

cout << "\nТип Money\n";

Set<money> sMoney1(3);

cin >> sMoney1;

cout << "sMoney1\n" << sMoney1 << endl;

Set<money> sMoney2 = sMoney1;

cout << "sMoney2\n " << sMoney2 << endl;

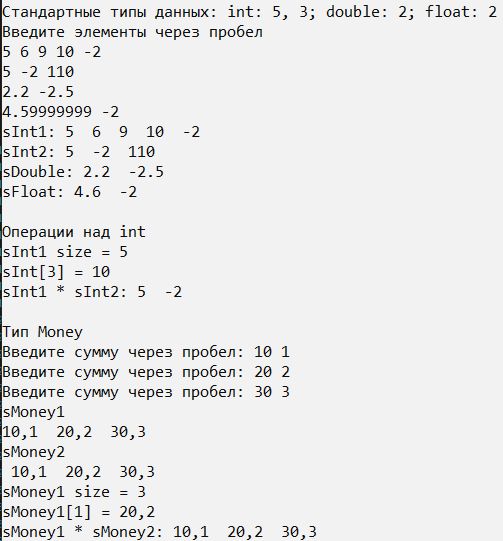
cout << "sMoney1 size = " << sMoney1() << endl;

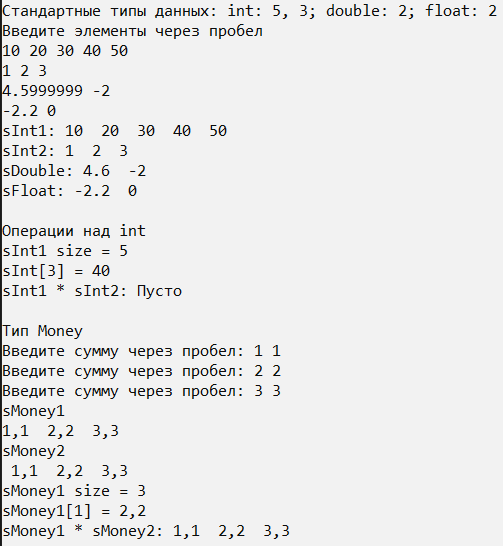
cout << "sMoney1[1] = " << sMoney1[1] << endl;

cout << "sMoney1 \* sMoney2: " << (sMoney1 \* sMoney2) << endl;

}

Результаты работы программы





Ответы на контрольные вопросы

1) Можно отделить алгоритм от конкретных типов данных, передавая тип в качестве параметра. Уменьшение кода, повышение читабельности.

2)

template <параметры\_шаблона>

загаловок\_функции

{ тело функции }

3)

template <параметры\_шаблона>

class Название

{ … }

4) Тип данных, который будет использоваться внутри

5) Можно использовать параметр в качестве типа для аргументов или переменных внутри класса, а также возвращаемого значения

6) template<название\_параметра [,следующие параметры])>

7) Нет, не имеет смысла

8) Можно использовать параметр в качестве типа для полей, возврата методов, и аргументов в методах

9) Нет, только те, в которых указан тип параметра

10) Нет, если не указать <>

11) Нет, не могут

12) Они будут иметь тот тип, который указан при инициализации объекта данного класса. Компилятор сам создаст перегруженную функцию данного типа

13) создание объектов определенного типа, примеи class\_name<int>, class\_name<Time>

14) На этапе компиляции