Лабораторная работа №20 «Шаблоны классов в С++»

Задание: МНОЖЕСТВО с элементами типа int. Реализовать операции: [] – доступа по индексу; () – определение размера вектора; - – разность множеств. 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

Цель работы: написать программу, которая демострирует работоспособность класса Plenty и взаимодействие этого класса с классом Money.

#include <iostream>

#include <ctime>

#include <vector>

#include <algorithm>

template<typename T> //Инстанцирования класса Plenty для работы с различными типами данных

class Plenty

{

private:

size\_t size;

std::vector<T> plenty;

public:

Plenty(size\_t size = 0)

{// Конструктор без параметров

this->size = size;

}

void insert(T number)

{// Функция для размещения чисел в множестве

if (size == 0)

{

plenty.push\_back(number);

size++;

}

if (coincidence(number) != false)

{

plenty.push\_back(number);

size++;

if (size > 1)

sort(plenty.begin(), plenty.end()); // Сортировка в порядке возрастания

}

}

~Plenty() {} // Деструктор

bool coincidence(T& number)

{// Функция для отслеживания: существует ли данное число в множестве

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

{

if (plenty[i] == number)

return false;

}

}

T Get\_size()

{// Геттер для поля size

return size;

}

const T& operator [] (int& index)

{// Перегрузка оператора []

return plenty[index];

}

const T& operator - (Plenty& other)

{// Перегрузка оператора -

size\_t other\_object\_size = other.Get\_size();

int in = 0;

int jn = 0;

int \_size = size;

if (size > other\_object\_size)

jn = size - other\_object\_size;

else if (size < other.Get\_size())

{

in = other\_object\_size - size;

\_size = other\_object\_size;

}

for (int i = 0; i < \_size - in; i++)

{

int count = 0;

for (int j = 0; j < \_size - jn; j++)

{

if (plenty[i] != other[j])

count++;

}

if (count == \_size - jn)

std::cout << plenty[i] << ' ';

}

return 0;

}

void print()

{// Вывод

for (auto x : plenty)

{

std::cout << x << ' ';

}

}

};

template<typename T>

class Money : public Plenty <T>

{

private:

long rubles;

int pennies;

public:

Money()

{// Конструктор без параметров

rubles = 1 + rand() % 1000;

pennies = rand() % 100;

}

~Money() {} // Деструктор

T Get\_rubles()

{// Геттер поля rubles

return rubles;

}

void print()

{// Функция вывода полей класса Money

std::cout << rubles << "," << pennies;

}

};

int main()

{

srand(time(NULL));

setlocale(LC\_ALL, "rus");

// 1. Создаем первый объект типа Plenty, инициализируем и выводим на экран

Plenty <int> plenty\_first;

std::cout << "Первое множество: ";

for (int i = 0; i < 2 + rand() % 8; i++)

{

plenty\_first.insert(rand() % 20);

}

plenty\_first.print();

std::cout << "\nSize: " << plenty\_first.Get\_size() << std::endl << std::endl; // Debug

// 2. Создаем второй объект типа Plenty, инициализируем и выводим на экран

Plenty <int> plenty\_second;

std::cout << "Второе множество: ";

for (int i = 0; i < 2 + rand() % 8; i++)

{

plenty\_second.insert(rand() % 20);

}

plenty\_second.print();

std::cout << "\nSize: " << plenty\_second.Get\_size() << std::endl;

// 3. Демонстрация работоспособности операции "-"(разности множеств)

std::cout << "\nРазность множества plenty\_first к plenty\_second: ";

plenty\_first - plenty\_second;

std::cout << std::endl;

// 4. Демонстрация перегрузки операции []

int n = rand() % plenty\_first.Get\_size();

std::cout << "\nplenty\_first[" << n << "]: " << plenty\_first[n] << std::endl << std::endl;

// 5.Демонстрация выполнения операция для контейнера, содержащий элементы пользовательского класса

std::cout << "Демонстрация выполнения операция для контейнера, содержащий элементы пользовательского класса: \n";

//5.1 Cоздаем 4 объекта типа Money и инициализируем

Money <int> money\_1;

Money <int> money\_\_1;

Money <int> money\_2;

Money <int> money\_\_2;

//5.2 Создаем объект plenty\_third типа Plenty и инициализируем его

Plenty <int> plenty\_third;

plenty\_third.insert(money\_1.Get\_rubles());

plenty\_third.insert(money\_\_1.Get\_rubles());

std::cout << "plenty\_third: ";

plenty\_third.print();

//5.3 Создаем объект plenty\_four типа Plenty и инициализируем его

Plenty <int> plenty\_four;

plenty\_four.insert(money\_2.Get\_rubles());

plenty\_four.insert(money\_\_2.Get\_rubles());

std::cout << "\nplenty\_four: ";

plenty\_four.print();

//5.4 Демонстриурем операцию "-"

std::cout << "\nРазность: ";

plenty\_third - plenty\_four;

std::cout << std::endl << "Демонстрация корректного вывода через запятую: ";

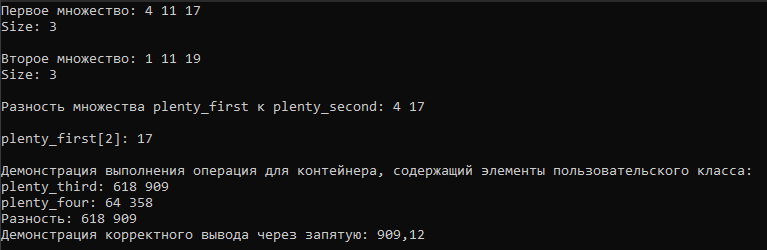
money\_1.print();

std::cout << std::endl;

return 0;

}

Рисунок 1. Результат выполнения



Постановка задачи:

1.Определить шаблон класса-контейнера (см. лабораторную работу №23).

Код с определением класса указан выше. Для инстанцирования данного класса используется template<typename T>, по указанию из 10 варианта данный класс работает с типом int тип для работы с классом задается при создании множества, например в 128 стр Plenty <int> plenty\_first; Этот класс имеет динамический массив vector и поле size с типом size\_t предназначенным для работы с размерами контейнеров.

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

В классе используется один деструктор и один конструктор, который инициализирует поле size, вектор инициализируется в функции insert

3.Перегрузить операции, указанные в варианте.

const T& operator [] (int& index)

{// Перегрузка оператора []

return plenty[index];

} Перегрузка операции получения значения элемента по индексу.

const T& operator - (Plenty& other)

{// Перегрузка оператора -

size\_t other\_object\_size = other.Get\_size();

int in = 0;

int jn = 0;

int \_size = size;

if (size > other\_object\_size)

jn = size - other\_object\_size;

else if (size < other.Get\_size())

{

in = other\_object\_size - size;

\_size = other\_object\_size;

}

for (int i = 0; i < \_size - in; i++)

{

int count = 0;

for (int j = 0; j < \_size - jn; j++)

{

if (plenty[i] != other[j])

count++;

}

if (count == \_size - jn)

std::cout << plenty[i] << ' ';

}

return 0;

} Перегрузка операции разности множеств “-“

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

Смотреть в 1-ом пункте Постановка задачи.

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

Код указан выше.

6.Реализовать пользовательский класс (см. лабораторную работу №22).

Класс Money наследует методы класса Plenty и так-же инстанцируется с помощью template<typename T> Класс Money имеет два поля 1 – long rubles для работы с рублями, 2 – int pennies для работы с копейками. Вывод данного класса осуществляется через метод print()

7.Перегрузить для пользовательского класса операции ввода-вывода.

Перегрузка данных операций в данном видео программы не предусмотрена, потому что все инициализируется с помощью ДСЧ.

8.Перегрузить операции необходимые для выполнения операций контейнерного класса.

Данный класс наследует перегруженные операции базового класса.

9.Инстанцировать шаблон для пользовательского класса.

Смотреть в 6 пункте Постановка задачи.

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

Код указан выше.

Содержание отчета

1.Постановка задачи (общая и конкретного варианта).

Описана в пункте Задание.

2.Описание параметризированного класса-контейнера.

Описано в Постановка задачи.

3.Определение компонентных функций.

Код указан выше.

4.Описание пользовательского класса и его компонентных функций

Описано в Постановка задачи.

5.Функция main().

Код указан выше.

6.Объяснение результатов работы программы.

Программа демонстрирует создание нескольких объектов класса множества и взаимодействие между ними, а также взаимодействие между классом Множество и классом Money.

Вывод по работе:

В ходе выполнения работы были получены навыки по созданию классов, а также знания о спецификаторах доступа, полях и методов, селекторах и модификаторах.