Министерство науки и высшего образования РФ

 Пензенский государственный университет Кафедра "Вычислительная техника"

Отчет по лабораторной работе №5

Работу выполнили студенты группы 23ВВВ3 бригады №9:

Полиневский Вадим

Кизым Иван

Проверили:

Кандидат т.н. Гудков А.А

Кандидат т.н. Евсеева И. Е

Пенза 2025

**Цель работы**: Изучить механизмы наследования в языке C++

**Задаие :** 9.

1. Доработать шаблонный класс из предыдущей лабораторной работы таким образом, чтобы в программе осуществлялась генерация и обработка исключительных ситуаций.

Исключительные ситуации генерируются:

1. В конструкторе с параметром при попытке создать контейнер больше максимального

размера;

1. В операции []
2. При попытке обратиться к элементу с номером меньше 0 или больше текущего размера контейнера ;
3. В операции +
4. При попытке добавить элемент с номером больше максимального размера;
5. В операции –
6. При попытке удалить элемент из пустого контейнера.
7. Создать программу, использующую механизм обработки исключительных ситуаций, позволяющую выполнять перевод вещественного числа из одной системы счисления в другую.

Варианты заданий:

1. Перевод числа из восемнадцатеричной системы счисления в двенадцатеричную.

**Описание практической части**:

Для выполнения программы было использовано два модуля: **lab5.cpp** с функцией main и файл заголовка **header.h.**

Генерация и обработка исключений выполнена для передачи пустого вектора в функии Remove\_Duplicates после перегрузки оператора + в классе Set; при условии превышения предусмотренного размера вектора в цикле при пользовательском вводе; а также при перегрузке оператора () при передаче пустого вектора.

Листинг:

**lab5.cpp:**

#include <iostream>

#include <vector>

#include "header.h"

using namespace std;

int main (){

cout << "lab4" << endl;

vector <int> plenty\_1;

vector <int> plenty\_2;

vector <int> plenty\_3;

//vector for exception withou initialization

vector <int> plenty\_4;

// Check exception for iterator > max\_size\_vector;

try {Enter\_Plenty(plenty\_1, 1);}

catch(const exception& ex){cerr << ex.what() << '\n';}

Enter\_Plenty(plenty\_2, 2);

Enter\_Plenty(plenty\_3, 3);

Operation\_Planty(plenty\_1, plenty\_2, plenty\_3, plenty\_4);

Conversion\_eighteen\_digit\_systeme\_to\_duodecimal();

}

**header.h:**

#pragma once

#include <stdexcept>

#include <string>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

template <typename Type>

class Set{

private:

Type \*plenty;

unsigned int Size\_Plenty;

public:

Set() : plenty(nullptr), Size\_Plenty(0) {}

Set(const Set& other) {

Size\_Plenty = other.Size\_Plenty;

if (Size\_Plenty > 0) {

plenty = new Type[Size\_Plenty];

for (unsigned int i = 0; i < Size\_Plenty; i++) {

plenty[i] = other.plenty[i];

}

} else {

plenty = nullptr;

}

}

Set& operator=(const Set& other) {

if (this != &other) { // Защита от самоприсваивания

delete[] plenty; // Освобождаем старую память

Size\_Plenty = other.Size\_Plenty;

if (Size\_Plenty > 0) {

plenty = new Type[Size\_Plenty];

for (unsigned int i = 0; i < Size\_Plenty; i++) {

plenty[i] = other.plenty[i];

}

} else {

plenty = nullptr;

}

}

return \*this;

}

unsigned int get\_size\_plenty() {

return Size\_Plenty;

}

Type\* get\_Plenty\_elements() {

return plenty;

}

void operator + (const Set& \_Set) {

if (this->Size\_Plenty == 0 || \_Set.Size\_Plenty == 0) {throw runtime\_error ("Exception: one of the vectors is empty\n");}

Set<Type> result;

// Добавляем все уникальные элементы из текущего множества

for (unsigned i = 0; i < Size\_Plenty; ++i) {

bool found = false;

for (unsigned j = 0; j < result.Size\_Plenty; ++j) {

if (plenty[i] == result.plenty[j]) {

found = true;

break;

}

}

if (!found) {

// Добавляем элемент в result

Type\* temp = new Type[result.Size\_Plenty + 1];

for (unsigned k = 0; k < result.Size\_Plenty; ++k) {

temp[k] = result.plenty[k];

}

temp[result.Size\_Plenty] = plenty[i];

delete[] result.plenty;

result.plenty = temp;

result.Size\_Plenty++;

}

}

// Добавляем все уникальные элементы из второго множества

for (unsigned i = 0; i < \_Set.Size\_Plenty; ++i) {

bool found = false;

for (unsigned j = 0; j < result.Size\_Plenty; ++j) {

if (\_Set.plenty[i] == result.plenty[j]) {

found = true;

break;

}

}

if (!found) {

// Добавляем элемент в result

Type\* temp = new Type[result.Size\_Plenty + 1];

for (unsigned k = 0; k < result.Size\_Plenty; ++k) {

temp[k] = result.plenty[k];

}

temp[result.Size\_Plenty] = \_Set.plenty[i];

delete[] result.plenty;

result.plenty = temp;

result.Size\_Plenty++;

}

}

// Заменяем текущее множество на результат

delete[] plenty;

plenty = result.plenty;

Size\_Plenty = result.Size\_Plenty;

// Чтобы result не удалил память в деструкторе

result.plenty = nullptr;

result.Size\_Plenty = 0;

}

void operator () (const vector <Type> & temp\_vector){

int max\_value =100;

if (temp\_vector.empty()) {throw runtime\_error ("Exception: this vector is empty\n");}

if (temp\_vector.size() > max\_value) {throw runtime\_error ("Exception: the size of this vector is larger than necessary\n");}

this->plenty = new Type[temp\_vector.size()];

Size\_Plenty = temp\_vector.size();

for (int i = 0; i < Size\_Plenty; i++) {

plenty[i] = temp\_vector[i];

}

}

~Set () {

delete[] plenty;

plenty = nullptr;

}

int operator <= (const Set& \_Set){

if (this->Size\_Plenty == \_Set.Size\_Plenty) {

for (int i = 0; i < Size\_Plenty; i++) {

if (this->plenty[i] != \_Set.plenty[i]) {return -1;}

}

return 0;

}

else {return -1;}

}

};

void Enter\_Plenty(vector <int>& temp\_vector, int number) {

const int count = 5;

int temp\_value;

cout << " ENTER " << number << " PLANTY:" << endl;

// size + 1 in for

if (number == 1){

for (int i = 0; i < count + 1; i++) {

if (i >= count) {throw runtime\_error("Exception: iterator >= size your vector");}

cout << i << ": ";

cin >> temp\_value;

temp\_vector.push\_back(temp\_value);

}

}

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

cout << i << ": ";

cin >> temp\_value;

temp\_vector.push\_back(temp\_value);

}

}

void Print\_Plenty(Set <int> \_Set , string Name\_Set){

int \*temp\_plenty;

unsigned int temp\_Size;

temp\_plenty = \_Set.get\_Plenty\_elements();

temp\_Size = \_Set.get\_size\_plenty();

cout << Name\_Set<< ": ";

for (int i = 0; i < temp\_Size; i++) {

cout << temp\_plenty[i] << ", ";

}

cout << endl;

}

void Operation\_Plenty(vector <int> temp\_1, vector<int> temp\_2, vector <int> temp\_3, vector <int> temp\_4){

Set <int> Set\_1, Set\_2, Set\_3, Set\_4, Set\_5;

vector <int> oferflow\_vector(102);

Set\_1(temp\_1);

Set\_2(temp\_2);

Set\_3(temp\_3);

try {Set\_4(temp\_4);}

catch(const std::exception& e){std::cerr << e.what() << '\n';}

try {Set\_5(oferflow\_vector);}

catch (const std::exception& e) {std::cerr << e.what() << '\n';}

Print\_Plenty(Set\_1, "Set\_1 operator () ");

Print\_Plenty(Set\_2, "Set\_2 operator () ");

Print\_Plenty(Set\_3, "Set\_3 operator () ");

Set\_1 + Set\_2;

Print\_Plenty (Set\_1, "Set\_1 operator (+) Set\_2");

// Exception: one of the vectors is empty\n

try {Set\_3 + Set\_4;}

catch(const std::exception& e) {std::cerr << e.what() << '\n';}

if (Set\_1 <= Set\_2 == 0) {cout << "sets are equal"<< endl;}

else {cout << "sets are not equal"<< endl;}

}

int eighteen\_digit\_to\_decimal(const string& eig){return stoi(eig, nullptr, 18);}

string decimal\_To\_doudecimal (int decimal) {

string base\_12 = "";

while (decimal > 0) {

int remainder = decimal % 12;

if (remainder < 10) base\_12 += (char)('0' +remainder);

else base\_12 += char('A' + (remainder - 10));

decimal /=12;

}

reverse(base\_12.begin(), base\_12.end());

if (base\_12.empty()) return "0";

else return base\_12;

}

void Conversion\_eighteen\_digit\_systeme\_to\_duodecimal(){

cout << endl;

string temp\_value;

cout << "ENTER VALUE IN HEXANDEMICAL SYSTEM: ";

cin >> temp\_value;

temp\_value = decimal\_To\_doudecimal(eighteen\_digit\_to\_decimal(temp\_value));

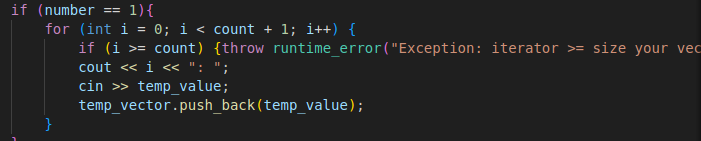
cout << "DUOCIMAL VALUE: " << temp\_value << endl;

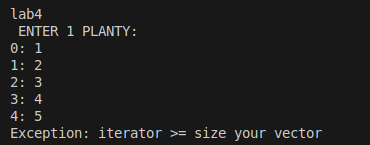
}

**Отладка:**

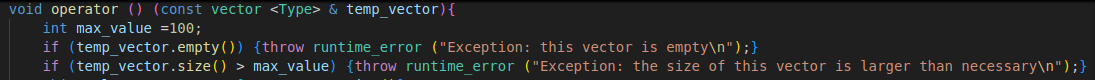
Обработка исключения: итаратор больше размера контейнера.

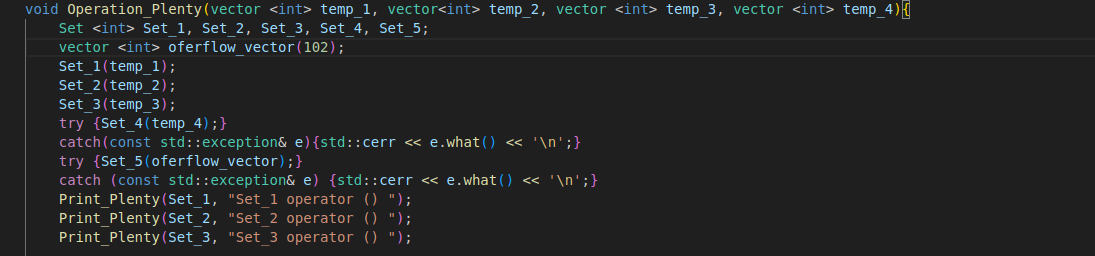


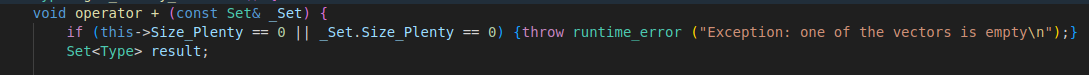




Обработка исключений: инициализация множества контейнером больше, чем допустимый размер(100) или пустым контейнером:



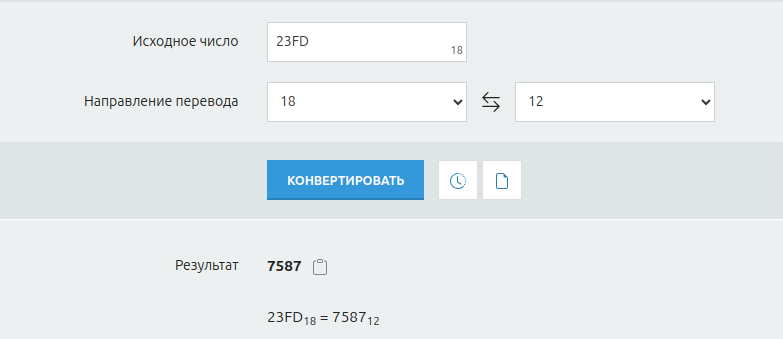
 Обработка исключения: передача пустого контейнера в оператор +:





Перевод числа из 16-ричной в 12-ричную систему.





**Terminal-Log:**

lab4

ENTER 1 PLANTY:

0: 1

1: 2

2: 3

3: 4

4: 5

Exception: iterator >= size your vector

ENTER 2 PLANTY:

0: 4

1: 5

2: 6

3: 7

4: 8

ENTER 3 PLANTY:

0: 9

1: 6

2: 5

3: 4

4: 3

Exception: this vector is empty

Exception: the size of this vector is larger than necessary

Set\_1 operator () : 1, 2, 3, 4, 5,

Set\_2 operator () : 4, 5, 6, 7, 8,

Set\_3 operator () : 9, 6, 5, 4, 3,

Set\_1 operator (+) Set\_2: 1, 2, 3, 4, 5, 6, 7, 8,

Exception: one of the vectors is empty

sets are not equal

ENTER VALUE IN HEXANDEMICAL SYSTEM: 23FD

DUOCIMAL VALUE: 7587