## **Лабораторная работа № 7**

**Контейнерные классы**

1. Постановка задачи

* Определить класс-контейнер.
* Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
* Перегрузить операции, указанные в варианте.
* Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
* Написать тестирующую программу, иллюстрирующую выполнение операций.

Класс-контейнер ВЕКТОР с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора;

\* число – умножает все элементы вектора на число;

-n – переход влево к элементу с номером n (с помощью класса-итератора).

1. Код программы

Vector.h

#pragma once

#include <iostream>

#include <exception>

using namespace std;

class Vector

{

private:

int\* a; //массив

int size; //размер (сколько эл-тов добавлено)

int max\_size; //максим размер (сколько можно добавить элементов)

const int MAX\_SIZE = 30;

public:

Vector(int max\_size);

Vector(const Vector& v);

~Vector();

int get\_size() { return size; }

Vector& operator=(const Vector& v);

int& operator[](int index);

int operator()();

Vector operator\*(int k);

Vector& operator+(int k);

Vector& operator--();

class iterator

{

int\* current;

public:

iterator(int\* p) : current(p){}

int& operator\*() { return \*current; }

int& operator\*() const { return \*current; }

bool operator==(const iterator& it) const { return current == it.current; }

bool operator!=(const iterator& it) const { return current != it.current; }

iterator& operator++() { ++current; return \*this; }

iterator operator++(int) { int\* tmp = current; ++current; return iterator(tmp); }

iterator& operator-(int n)

{

current = current - n;

return \*this;

}

friend class Vector;

};

iterator begin() const { return iterator(a); }

iterator end() const { return iterator(a+size); }

};

Vector.cpp

#include "Vector.h"

Vector::Vector(int max\_size)

{

if (max\_size > MAX\_SIZE)

throw exception("размер больше максимального");

size = 0;

this->max\_size = max\_size;

a = new int[max\_size];

}

Vector::Vector(const Vector& v)

{

size = v.size;

max\_size = v.max\_size;

a = new int[max\_size];

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

{

a[i] = v.a[i];

}

}

Vector::~Vector()

{

delete[] a;

}

Vector& Vector::operator=(const Vector& v)

{

if(this != &v)

{

delete[] a;

a = new int[v.max\_size];

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

a[i] = v.a[i];

size = v.size;

max\_size = v.max\_size;

}

return \*this;

}

int& Vector::operator[](int index)

{

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

throw exception("выход за границы вектора");

return a[index];

}

int Vector::operator()()

{

return size;

}

Vector Vector::operator\*(int k)

{

Vector res(\*this);

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

{

res.a[i] = res.a[i] \* k;

}

return res;

}

Vector& Vector::operator+(int k)

{

if (size == max\_size)

throw exception("вектор полный");

a[size] = k;

size++;

return \*this;

}

Vector& Vector::operator--()

{

if (size == 0)

throw exception("вектор пустой");

for (int i = 0; i < size-1; i++)

{

a[i] = a[i + 1];

}

size--;

return \*this;

}

Main.cpp

#include <iostream>

#include "Vector.h"

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Размер вектора = ";

int size;

cin >> size;

try

{

Vector v(size);

do

{

cout << "1.Добавить элемент" << endl;

cout << "2.Удалить элемент из начала вектора" << endl;

cout << "3.Получить элемент по индексу" << endl;

cout << "4.Вывести элементы" << endl;

cout << "5.Умножение вектора на число" << endl;

cout << "6.Получение элемента с конца вектора" << endl;

cout << "0.Выход" << endl;

int oper;

cin >> oper;

if (oper == 0)

break;

try

{

switch (oper)

{

case 1:

{

cout << "Введите значение элемента: ";

int element;

cin >> element;

v = v + element;

break;

}

case 2:

{

--v;

break;

}

case 3:

{

cout << "Введите индекс элемента: ";

int index;

cin >> index;

cout << v[index] << endl;

break;

}

case 4:

{

Vector::iterator start = v.begin();

Vector::iterator end = v.end();

while (start != end)

{

cout << \*start << " ";

start++;

}

cout << endl;

break;

}

case 5:

{

cout << "Введите число: ";

int k;

cin >> k;

v = v \* k;

break;

}

case 6:

{

cout << "Введите номер: ";

int n;

cin >> n;

Vector::iterator end = v.end();

end = end - n;

cout << \*end << endl;

break;

}

default:

break;

}

}

catch (const exception & exc)

{

cout << exc.what() << endl;

}

}

while (true);

}

catch (const exception& exc)

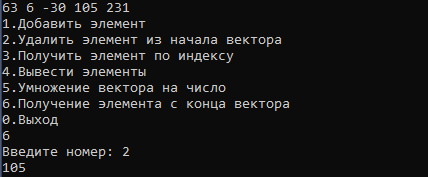
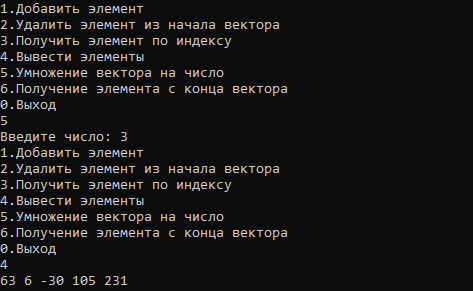
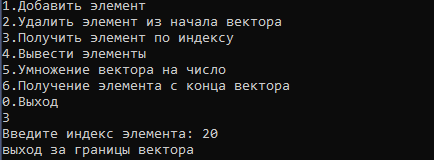
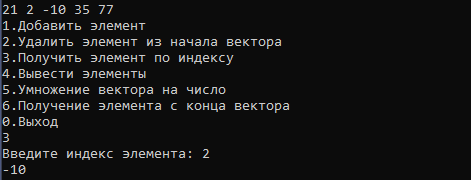
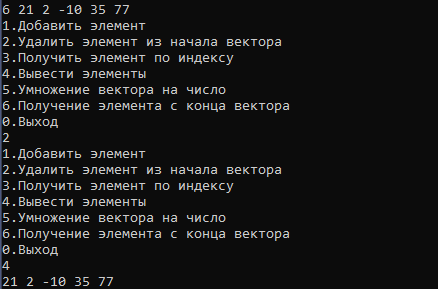
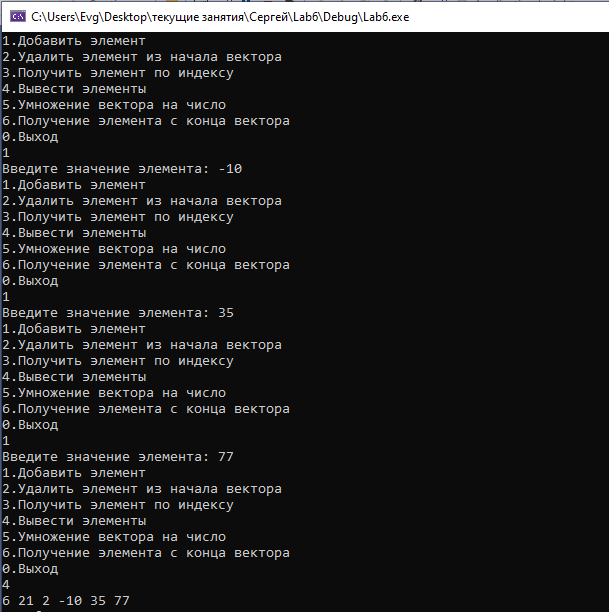
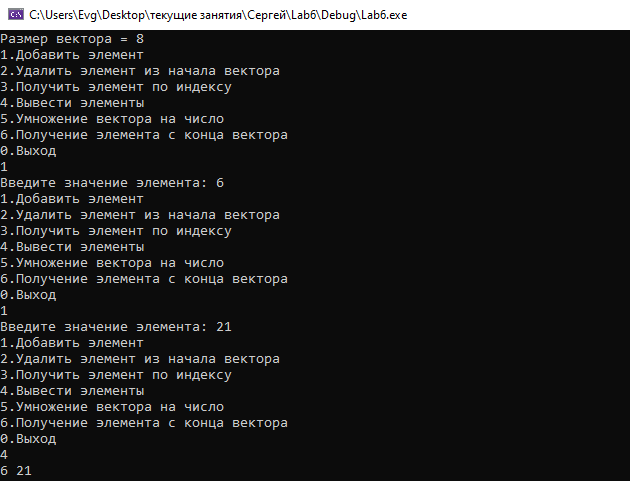
{

cout << exc.what() << endl;

}

}

1. Результат выполнения



## **Лабораторная работа № 8**

## **Базы данных**

1. Постановка задачи

* Написать приложение для работы с простой базой данных, хранящей информацию об объекте на внешнем носителе. Приложение должно выполнять следующие функции:
* Создание базы данных, содержащей записи указанного формата.
* Просмотр базы данных.
* Удаление элементов из базы данных (по ключу/ по номеру).
* Корректировка элементов в базе данных (по ключу / по номеру).
* Добавление элементов в базу данных (в начало / в конец/ с заданным номером).
* Выполнение задания, указанного в варианте.

Типизированный файл, имеет следующую структуру: Автор, Название, Год издания, Издательство. Программа должна

* Добавлять, удалять, корректировать, позволять просматривать записи файла.
* Выдавать по запросу пользователя: список литературы, указанного пользователем автора;
* Список литературы, изданной в указанный пользователем период.

1. Код программы

Book.h

#pragma once

#include<iostream>

#include <string>

using namespace std;

class book

{

private:

string name;

string author;

int year;

string publish;

bool deleted;

public:

book(){}

book(string name, string author, int year, string publish);

void set\_name(string name) { this->name = name; }

string get\_name() { return this->name; }

void set\_author(string author) { this->author = author; }

string get\_author() { return this->author; }

void set\_year(int year) { this->year = year; }

int get\_year() { return this->year; }

void set\_publish(string publish) { this->publish = publish; }

string get\_publish() { return this->publish; }

void set\_deleted(bool deleted) { this->deleted = deleted; }

bool get\_deleted() { return this->deleted; }

};

Book.cpp

#include "book.h"

book::book(string name, string author, int year, string publish)

{

set\_name(name);

set\_author(author);

set\_year(year);

set\_publish(publish);

set\_deleted(false);

}

Main.cpp

#include <iostream>

#include "book.h"

#include <list>

#include <fstream>

#include <Windows.h>

#include <iomanip>

using namespace std;

book\* create()

{

cin.ignore();

string name;

cout << "Введите название: ";

getline(cin, name);

string author;

cout << "Введите автора: ";

getline(cin, author);

string publish;

cout << "Введите издательство: ";

getline(cin, publish);

int year;

cout << "Введите год: ";

cin >> year;

book\* b = new book(name, author, year, publish);

return b;

}

void show(const list<book\*>& books)

{

cout << setw(5) << "№" << setw(20) << "Название" << setw(30) << "Автор" << setw(10) << "Год" << setw(25) << "Издательство" << endl;

auto start = books.begin();

int i = 1;

while (start != books.end())

{

book\* b = \*start;

if (!b->get\_deleted())

{

cout << setw(5) << i << setw(20) << b->get\_name() << setw(30) << b->get\_author() << setw(10)

<< b->get\_year() << setw(25) << b->get\_publish() << endl;

i++;

}

start++;

}

}

//поиск по ключу

list<book\*>::const\_iterator find\_book(const list<book\*>& books, string name, string author)

{

auto start = books.begin();

while (start != books.end())

{

book\* b = \*start;

if (b->get\_name().compare(name) == 0 && b->get\_author().compare(author) == 0)

return start;

start++;

}

return books.end();

}

//поиск по номеру

list<book\*>::const\_iterator find\_book(const list<book\*>& books, int number)

{

auto start = books.begin();

int i = 1;

while (start != books.end())

{

book\* b = \*start;

if (i == number)

return start;

start++;

if (!b->get\_deleted())

i++;

}

return books.end();

}

void edit(book\* b)

{

cout << "Что редактировать? 1.Название 2.Автор 3.Год 4.Издательство" << endl;

int oper;

cin >> oper;

cin.ignore();

switch (oper)

{

case 1:

{

cout << "Введите новое название: ";

string name;

getline(cin, name);

b->set\_name(name);

break;

}

case 2:

{

cout << "Введите нового автора: ";

string author;

getline(cin, author);

b->set\_author(author);

break;

}

case 3:

{

cout << "Введите новый год: ";

int year;

cin >> year;

b->set\_year(year);

break;

}

case 4:

{

cout << "Введите новое издательство: ";

string publish;

getline(cin, publish);

b->set\_publish(publish);

break;

}

}

}

bool edit(list<book\*>& books)

{

cout << "1.По ключу" << endl;

cout << "2.По номеру" << endl;

int choose;

cin >> choose;

list<book\*>::const\_iterator find\_b = books.end();

switch (choose)

{

case 1:

{

cin.ignore();

string name, author;

cout << "Введите название: ";

getline(cin, name);

cout << "Введите автора: ";

getline(cin, author);

find\_b = find\_book(books, name, author);

break;

}

case 2:

{

int number;

cout << "Введите номер: ";

cin >> number;

find\_b = find\_book(books, number);

break;

}

}

if (find\_b != books.end())

{

edit(\*find\_b);

return true;

}

else

return false;

}

bool remove\_from\_list(list<book\*>& books)

{

int count\_del\_books = 0;

auto start = books.begin();

while (start != books.end())

{

book\* b = \*start;

if (b->get\_deleted())

count\_del\_books++;

start++;

}

if (count\_del\_books > books.size() / 2)

{

start = books.begin();

while (start != books.end())

{

book\* b = \*start;

if (b->get\_deleted())

{

books.erase(start++);

}

else

start++;

}

return true;

}

return false;

}

bool remove(list<book\*>& books, book\*\* remove\_book)

{

cout << "1.По ключу" << endl;

cout << "2.По номеру" << endl;

int choose;

cin >> choose;

list<book\*>::const\_iterator find\_b = books.end();

switch (choose)

{

case 1:

{

cin.ignore();

string name, author;

cout << "Введите название: ";

getline(cin, name);

cout << "Введите автора: ";

getline(cin, author);

find\_b = find\_book(books, name, author);

break;

}

case 2:

{

int number;

cout << "Введите номер: ";

cin >> number;

find\_b = find\_book(books, number);

break;

}

}

if (find\_b != books.end())

{

\*remove\_book = \*find\_b;

(\*find\_b)->set\_deleted(true);

if (remove\_from\_list(books))

\*remove\_book = nullptr;

return true;

}

else

return false;

}

void show\_by\_author(const list<book\*>& books, string author)

{

cout << setw(5) << "№" << setw(20) << "Название" << setw(30) << "Автор" << setw(10) << "Год" << setw(25) << "Издательство" << endl;

auto start = books.begin();

int i = 1;

while (start != books.end())

{

book\* b = \*start;

if (!b->get\_deleted() && b->get\_author().compare(author) == 0)

{

cout << setw(5) << i << setw(20) << b->get\_name() << setw(30) << b->get\_author() << setw(10)

<< b->get\_year() << setw(25) << b->get\_publish() << endl;

i++;

}

start++;

}

}

void show\_by\_period(const list<book\*>& books, int start\_year, int end\_year)

{

cout << setw(5) << "№" << setw(20) << "Название" << setw(30) << "Автор" << setw(10) << "Год" << setw(25) << "Издательство" << endl;

auto start = books.begin();

int i = 1;

while (start != books.end())

{

book\* b = \*start;

if (!b->get\_deleted() && b->get\_year() >= start\_year && b->get\_year() <= end\_year)

{

cout << setw(5) << i << setw(20) << b->get\_name() << setw(30) << b->get\_author() << setw(10)

<< b->get\_year() << setw(25) << b->get\_publish() << endl;

i++;

}

start++;

}

}

void add\_book(list<book\*>& books)

{

cout << "Куда добавить? 1.Начало 2.Конец 3.Заданная позиция" << endl;

int choose;

cin >> choose;

int pos;

if (choose == 1)

pos = 0;

else if (choose == 2)

pos = books.size();

else

{

cout << "Введите позицию: ";

cin >> pos;

}

book\* b = create();

auto it = books.begin();

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

it++;

books.insert(it, b);

}

void save\_in\_file(const list<book\*>& books)

{

ofstream file("books.bin", ios::binary);

auto start = books.begin();

while (start != books.end())

{

book\* b = \*start;

file.write((char\*)b, sizeof(\*b));

start++;

}

file.close();

}

void load\_file(list<book\*>& books)

{

ifstream file("books.bin", ios::binary);

if (file.is\_open())

{

book\* b = new book();

while (file.read((char\*)b, sizeof(book)))

{

books.push\_back(b);

b = new book();

}

file.close();

}

}

bool reestablish\_book(list<book\*>& books, book\* remove\_book)

{

if (remove\_book == nullptr)

return false;

auto start = books.begin();

while (start != books.end())

{

book\* b = \*start;

if (b == remove\_book)

{

b->set\_deleted(false);

return true;

}

start++;

}

return false;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

list<book\*> books;

load\_file(books);

book\* last\_remove\_book = nullptr;

int oper;

do

{

cout << "1.Добавление" << endl;

cout << "2.Удаление" << endl;

cout << "3.Редактирование" << endl;

cout << "4.Просмотр" << endl;

cout << "5.Поиск по автору" << endl;

cout << "6.Поиск по периоду" << endl;

cout << "7.Сохранить изменения" << endl;

cout << "8.Отмена удаления" << endl;

cout << "0.Выход" << endl;

cin >> oper;

switch (oper)

{

case 1:

{

add\_book(books);

break;

}

case 2:

{

bool res = remove(books, &last\_remove\_book);

if (res)

cout << "Удаление успешно завершено" << endl;

else

cout << "Книга не найдена" << endl;

break;

}

case 3:

{

bool res = edit(books);

if (res)

cout << "Редактирование успешно завершено" << endl;

else

cout << "Книга не найдена" << endl;

break;

}

case 4:

{

show(books);

break;

}

case 5:

{

cout << "Введите автора: ";

string author;

cin.ignore();

getline(cin, author);

show\_by\_author(books, author);

break;

}

case 6:

{

int start\_year, end\_year;

cout << "Начало периода: ";

cin >> start\_year;

cout << "Конец периода: ";

cin >> end\_year;

show\_by\_period(books, start\_year, end\_year);

break;

}

case 7:

{

save\_in\_file(books);

break;

}

case 8:

{

bool res = reestablish\_book(books, last\_remove\_book);

if (res)

cout << "Последнее удаление отменено" << endl;

else

cout << "Отмена невозможна" << endl;

}

default:

break;

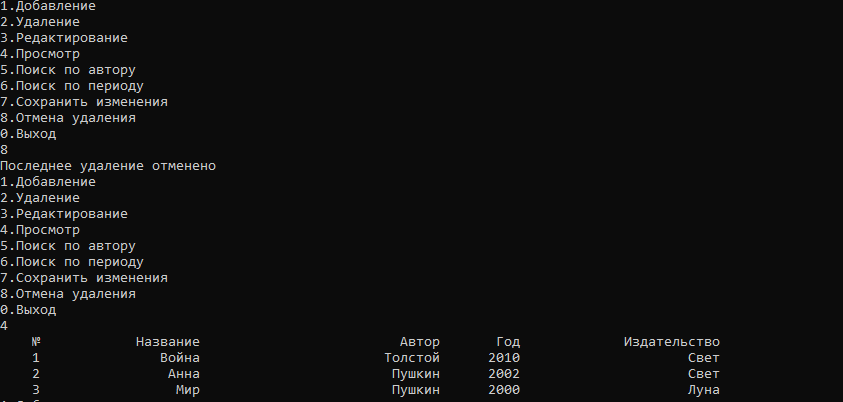
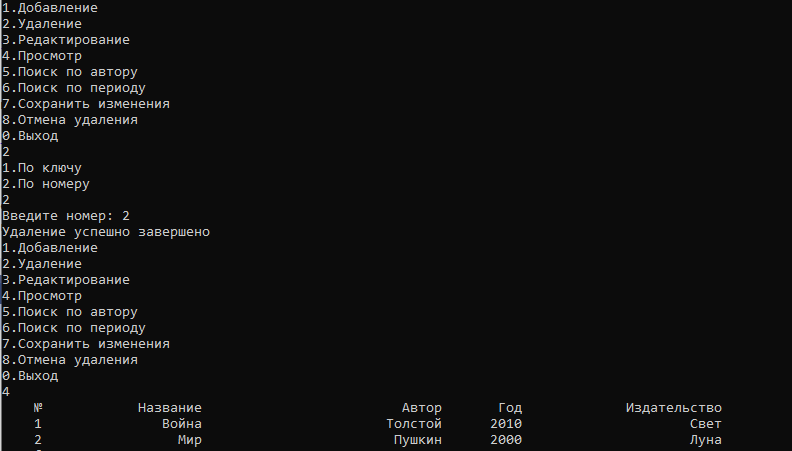
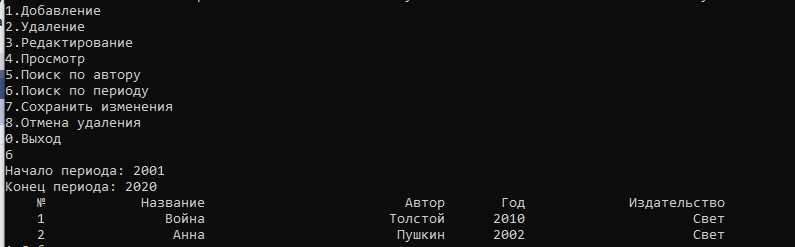
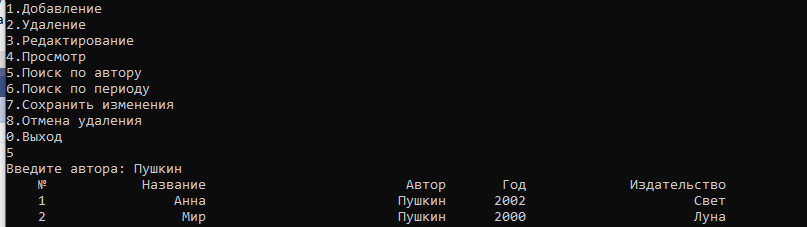
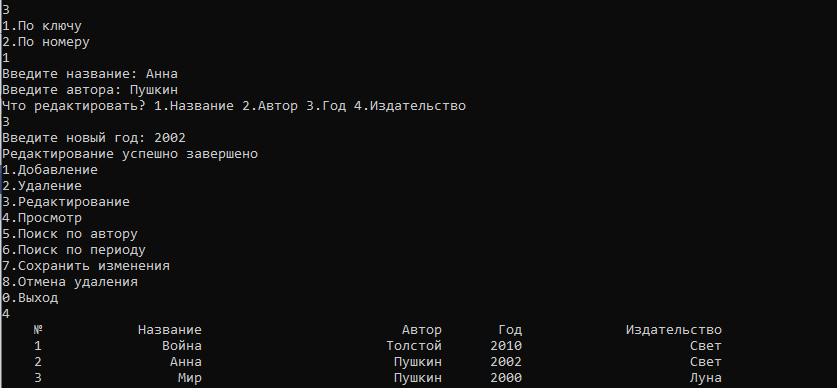
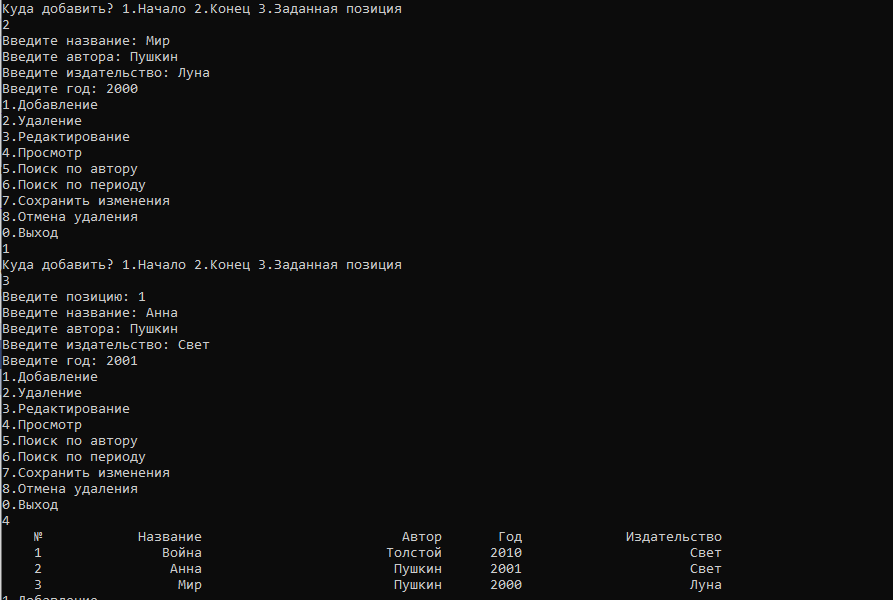
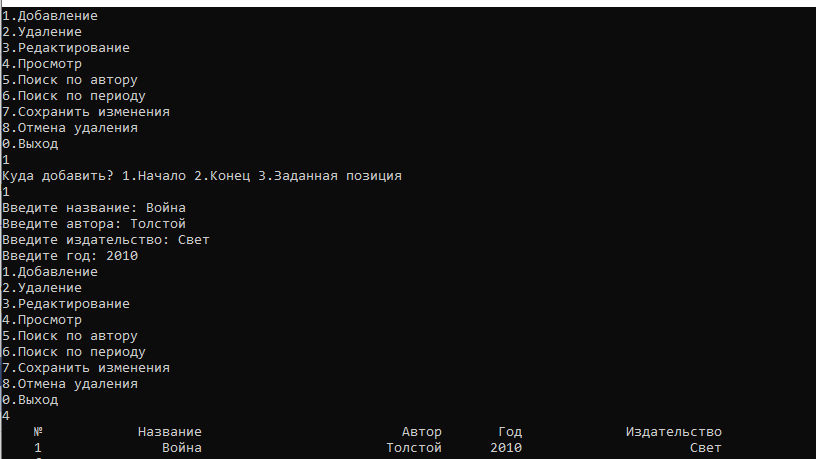
}

}

while (oper != 0);

}

1. Результаты выполнения



## **Лабораторная работа № 9**

## **Поиск данных с помощью хеш-таблиц**

1. Постановка задачи

* Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.
* Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хеш-таблицу.
* Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90 элементов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | ФИО, №паспорта, адрес | №паспорта | H(k)= [M (kAmod1)], 0<A<1, mod1 – получение дробной части, [] – получение целой части | Метод цепочек |

1. Код программы

Person.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class person

{

private:

string fio;

int number\_passport;

string address;

public:

person(string fio, int passport, string address) : fio(fio), number\_passport(passport), address(address){}

string get\_fio() { return fio; }

void set\_fio(string fio) { this->fio = fio; }

int get\_number\_passport() { return number\_passport; }

void set\_number\_passport(int number\_passport) { this->number\_passport = number\_passport; }

string get\_address() { return address; }

void set\_address(string address) { this->address = address; }

};

Hashtable.h

#pragma once

#include "person.h"

class hashtable

{

private:

class node

{

public:

int key;

person\* p;

node\* next;

};

int max\_size;

node\*\* table;

int hash\_code(int key);

public:

hashtable(int size);

~hashtable();

void add(int key, person\* p);

person\* find(int key);

bool remove(int key);

int count\_collision();

void show();

};

Hashtable.cpp

#include "hashtable.h"

int hashtable::hash\_code(int key)

{

double a = 0.7;

double ka = key \* a;

double drob\_part = ka - (int)ka;

return (int)(drob\_part \* max\_size);

}

hashtable::hashtable(int size)

{

max\_size = size;

table = new node \* [max\_size];

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

{

table[i] = nullptr;

}

}

hashtable::~hashtable()

{

delete[] table;

}

void hashtable::add(int key, person\* p)

{

int code = hash\_code(key);

node\* point = new node();

point->p = p;

point->key = key;

point->next = nullptr;

if (table[code] == nullptr)

{

table[code] = point;

}

else

{

point->next = table[code];

table[code] = point;

}

}

person\* hashtable::find(int key)

{

int code = hash\_code(key);

node\* current = table[code];

while (current != nullptr)

{

if (current->key == key)

return current->p;

current = current->next;

}

return nullptr;

}

bool hashtable::remove(int key)

{

int code = hash\_code(key);

node\* current = table[code];

if (current != nullptr && current->key == key)

{

table[code] = table[code]->next;

delete current;

return true;

}

else

{

while (current->next != nullptr)

{

if (current->next->key == key)

{

node\* temp = current->next;

current->next = current->next->next;

delete temp;

return true;

}

current = current->next;

}

}

return false;

}

int hashtable::count\_collision()

{

int count = 0;

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

{

node\* current = table[i];

if (current != nullptr)

{

while (current != nullptr)

{

count++;

current = current->next;

}

count--;

}

}

return count;

}

void hashtable::show()

{

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

{

node\* current = table[i];

if (current != nullptr)

{

while (current != nullptr)

{

cout << "ФИО: " << current->p->get\_fio() << endl;

cout << "Номер пасспорта: " << current->p->get\_number\_passport() << endl;

cout << "Адрес: " << current->p->get\_address() << endl << endl;

current = current->next;

}

}

}

}

Main.cpp

#include <iostream>

#include <vector>

#include <ctime>

#include <Windows.h>

#include "hashtable.h"

void show\_person(person\* p)

{

cout << "ФИО: " << p->get\_fio() << endl;

cout << "Номер пасспорта: " << p->get\_number\_passport() << endl;

cout << "Адрес: " << p->get\_address() << endl;

}

int main()

{

srand(time(0));

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

vector<string> surnames = { "Иванов","Петров","Сидоров","Тесла","Маск","Эйнштейн","Ньютон","Гук","Кюри","Сталин","Ленин","Маркс" };

vector<string> names = { "Иван","Петр","Сидор","Никола","Илон","Альберт","Исаак","Роберт","Мария","Иосиф","Владимир","Карл","Алексей","Михаил","Дмитрий" };

vector<string> patronymics = { "Иванович","Петрович","Сидорович","Николаевич","Илонович","Альбертович","Исаакович","Робертович","Маркович","Иосифович","Владимирович","Карлович","Алексеевич","Михаилович","Дмитриевич" };

vector<string> addresses = { "Мира 10", "Ленина 50", "Ленина 20", "Декабристов 45", "Попова 3" };

hashtable\* table = nullptr;

int oper;

do

{

cout << "1.Создание хэш-таблицы" << endl;

cout << "2.Заполнение хэш-таблицы" << endl;

cout << "3.Поиск по ключу" << endl;

cout << "4.Определение кол-ва коллизий" << endl;

cout << "5.Просмотр хэш-таблицы" << endl;

cout << "6.Удаление по ключу" << endl;

cout << "7.Редактирование по ключу" << endl;

cout << "0.Выход" << endl;

cin >> oper;

switch (oper)

{

case 1:

{

int size;

cout << "Размер = ";

cin >> size;

table = new hashtable(size);

break;

}

case 2:

{

int count;

cout << "Кол-во эл-тов = ";

cin >> count;

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

{

string fio = surnames[rand() % surnames.size()] + " " + names[rand() % names.size()] + " " + patronymics[rand() % patronymics.size()];

int passport = rand() + 100000;

string address = addresses[rand() % addresses.size()];

person\* p = new person(fio, passport, address);

table->add(passport, p);

}

break;

}

case 3:

{

int passport;

cout << "Введите ключ для поиска: ";

cin >> passport;

person\* p = table->find(passport);

if (p == nullptr)

cout << "Элемент не найден" << endl;

else

{

show\_person(p);

}

break;

}

case 4:

{

int count = table->count\_collision();

cout << "Кол-во коллизий = " << count << endl;

break;

}

case 5:

{

table->show();

break;

}

case 6:

{

int passport;

cout << "Введите номер пасспорта: ";

cin >> passport;

bool res = table->remove(passport);

if (res)

cout << "Удаление успешно завершено" << endl;

else

cout << "Элемент не найден" << endl;

break;

}

case 7:

{

int passport;

cout << "Введите номер пасспорта: ";

cin >> passport;

person\* p = table->find(passport);

if (p == nullptr)

cout << "Элемент не найден" << endl;

else

{

cout << "Что поменять? 1.ФИО 2.Адрес" << endl;

int choose;

cin >> choose;

cin.ignore();

switch (choose)

{

case 1:

{

string fio;

cout << "Введите новое ФИО: ";

getline(cin, fio);

p->set\_fio(fio);

break;

}

case 2:

{

string address;

cout << "Введите новый адрес: ";

getline(cin, address);

p->set\_address(address);

break;

}

}

}

}

default:

break;

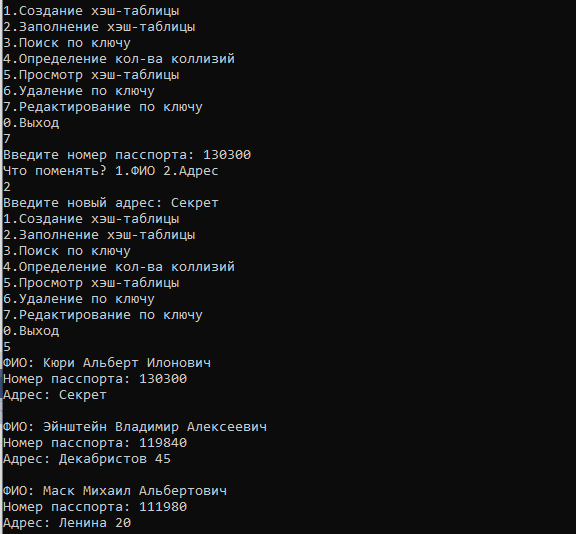
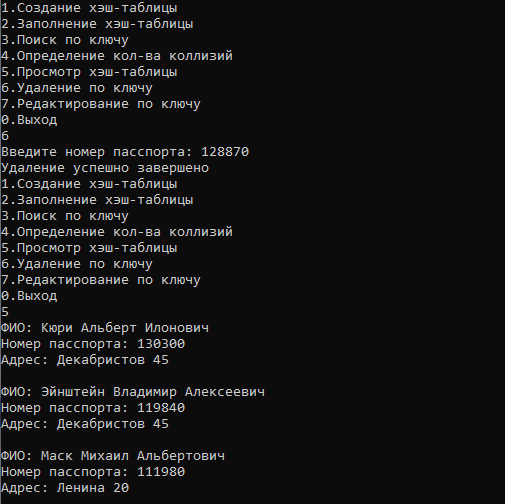
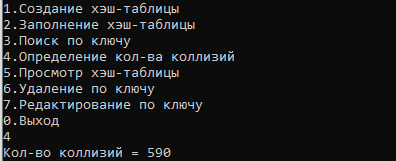
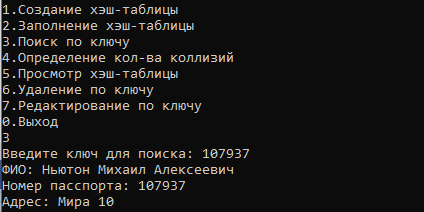
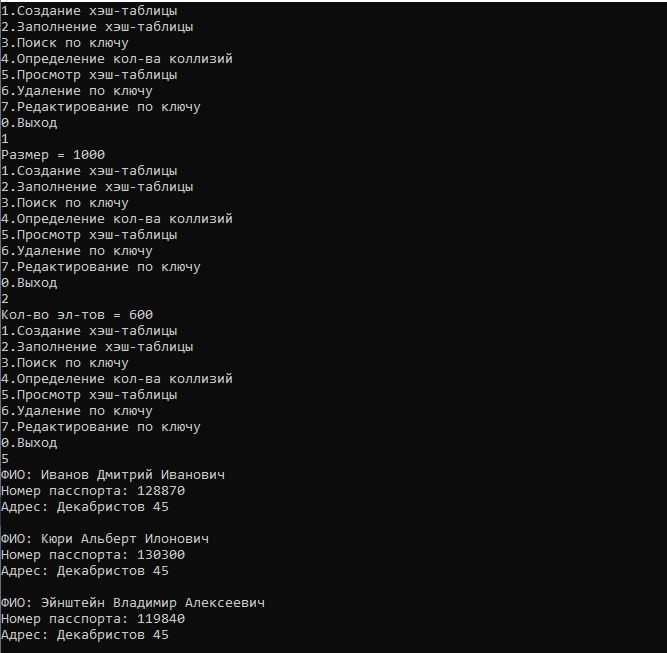
}

}

while (oper != 0);

}

1. Результаты работы



## **Лабораторная работа № 10**

## **Бинарные деревья**

1. Постановка задачи

* Сформировать идеально сбалансированное бинарное дерево, тип информационного поля указан в варианте.
* Распечатать полученное дерево.
* Выполнить обработку дерева в соответствии с заданием, вывести полученный результат.
* Преобразовать идеально сбалансированное дерево в дерево поиска.
* Распечатать полученное дерево.

Тип информационного поля double. Найти минимальный элемент в дереве.

1. Код программы

Point.h

#pragma once

#include<iostream>

using namespace std;

struct point

{

double data;

point\* left;

point\* right;

};

point\* create\_point(double data)

{

point\* p = new point();

p->data = data;

p->left = nullptr;

p->right = nullptr;

return p;

}

point\* build\_ideal\_tree(int n)

{

if (n == 0)

return NULL;

point\* p = create\_point(rand() % 100);

int n1 = n / 2;

int n2 = n - n1 - 1;

p->left = build\_ideal\_tree(n1);

p->right = build\_ideal\_tree(n2);

return p;

}

void print\_tree(point\* root, int level)

{

if (root != nullptr)

{

print\_tree(root->right, level + 1);

for (int i = 0; i < 5\*level; i++)

cout << " ";

cout << root->data << endl;

print\_tree(root->left, level + 1);

}

}

void find\_min(point\* root, double& min)

{

if (root != nullptr)

{

if (root->data < min)

min = root->data;

find\_min(root->left, min);

find\_min(root->right, min);

}

}

double find\_min\_find\_tree(point\* root)

{

while (root->left != nullptr)

root = root->left;

return root->data;

}

void destroy(point\* root)

{

if (root != NULL)

{

destroy(root->left);

destroy(root->right);

delete root;

}

}

void add(point\*\* root, double val)

{

if ((\*root) == NULL)

{

(\*root) = create\_point(val);

return;

}

if (val > (\*root)->data)

add(&(\*root)->right, val);

else if (val < (\*root)->data)

add(&(\*root)->left, val);

}

void build\_find\_tree(point\* root, point\*\* new\_root)

{

if (root != nullptr)

{

add(new\_root, root->data);

build\_find\_tree(root->left, new\_root);

build\_find\_tree(root->right, new\_root);

}

}

Main.cpp

#include <iostream>

#include "point.h"

#include <ctime>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

int choose;

point\* root = nullptr;

bool is\_tree\_find = false;

do

{

cout << "0.Выход" << endl;

cout << "1.Сформировать идеально сбалансированное бинарное дерево" << endl;

cout << "2.Распечатать дерево" << endl;

cout << "3.Найти минимальный элемент в дереве" << endl;

cout << "4.Преобразовать идеально сбалансированное дерево в дерево поиска" << endl;

cin >> choose;

switch (choose)

{

case 1:

{

int count;

cout << "Кол-во эл-тов = ";

cin >> count;

if (root != nullptr)

destroy(root);

is\_tree\_find = false;

root = build\_ideal\_tree(count);

break;

}

case 2:

{

if (root != nullptr)

print\_tree(root, 0);

else

cout << "Сначала сформируйте дерево!" << endl;

break;

}

case 3:

{

if (root != nullptr)

{

double min;

if (is\_tree\_find == false)

{

min = root->data;

find\_min(root, min);

}

else

{

min = find\_min\_find\_tree(root);

}

cout << "Минимальный элемент = " << min << endl;

}

else

cout << "Сначала сформируйте дерево!" << endl;

break;

}

case 4:

{

if (root != nullptr)

{

point\* new\_root = nullptr;

build\_find\_tree(root, &new\_root);

destroy(root);

root = new\_root;

is\_tree\_find = true;

}

else

cout << "Сначала сформируйте дерево!" << endl;

break;

}

case 0:

{

destroy(root);

break;

}

default:

cout << "Неверный пункт меню" << endl;

break;

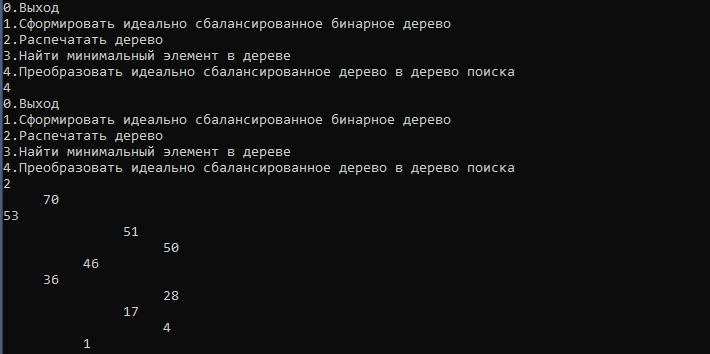
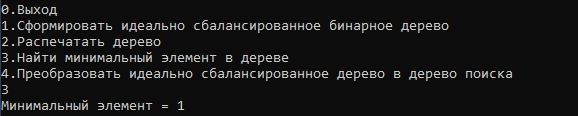
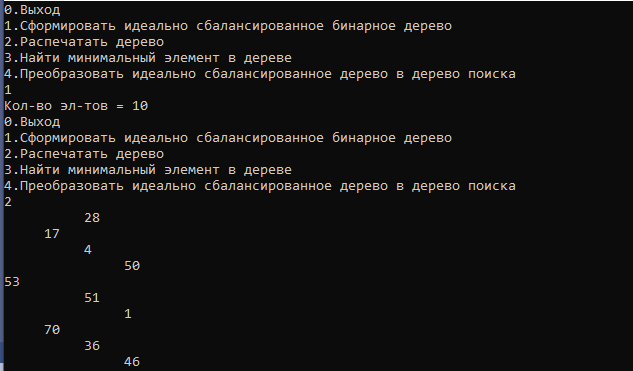
}

}

while (choose != 0);

}

1. Результаты работы



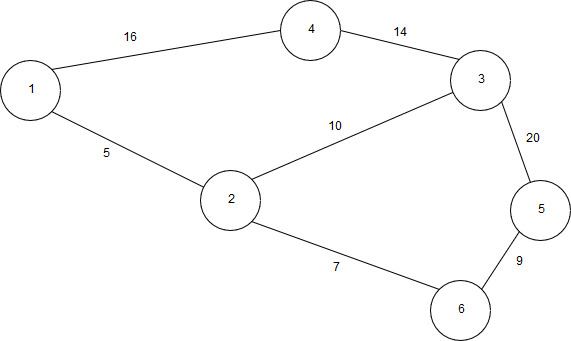
## **Лабораторная работа № 11**

## **Введение в теорию графов. Алгоритмы Дейкстры и Флойда**

1.Постановка задачи

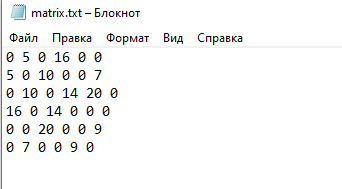
Реализовать граф, а также алгоритм Дейкстры, выполнив все необходимые действия.

Выполнение начать с вершины 2.



2. Код программы

Матрицу расстояний считываем из файла



#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

using namespace std;

#define SIZE 6

int main()

{

setlocale(LC\_ALL, "rus");

int a[SIZE][SIZE]; // матрица связей

int d[SIZE]; // минимальное расстояние

int v[SIZE]; // посещенные вершины

int temp, minindex, min;

int begin\_index = 1;

ifstream file("matrix.txt");

// Инициализация матрицы связей

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

{

for (int j = 0; j < SIZE; j++) {

file >> a[i][j];

}

}

file.close();

// Вывод матрицы связей

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

{

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

cout << a[i][j] << "\t";

cout << endl;

}

//Инициализация вершин и расстояний

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

{

d[i] = 10000;

v[i] = 1;

}

d[begin\_index] = 0;

// Шаг алгоритма

do {

minindex = 10000;

min = 10000;

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

{ // Если вершину ещё не обошли и вес меньше min

if ((v[i] == 1) && (d[i] < min))

{ // Переприсваиваем значения

min = d[i];

minindex = i;

}

}

// Добавляем найденный минимальный вес

// к текущему весу вершины

// и сравниваем с текущим минимальным весом вершины

if (minindex != 10000)

{

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

{

if (a[minindex][i] > 0)

{

temp = min + a[minindex][i];

if (temp < d[i])

{

d[i] = temp;

}

}

}

v[minindex] = 0;

}

} while (minindex < 10000);

// Вывод кратчайших расстояний до вершин

cout << "\nКратчайшие расстояния до вершин:" << endl;

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

cout << d[i] << "\t";

// Восстановление пути

int ver[SIZE]; // массив посещенных вершин

int endPoint;

cout << "\nВведите номер вершины до которой найти кратчайший путь ";

cin >> endPoint;

int end = endPoint - 1; // индекс конечной вершины = 5 - 1

ver[0] = end + 1; // начальный элемент - конечная вершина

int k = 1; // индекс предыдущей вершины

int weight = d[end]; // вес конечной вершины

while (end != begin\_index) // пока не дошли до начальной вершины

{

for (int i = 0; i < SIZE; i++) // просматриваем все вершины

if (a[i][end] != 0) // если связь есть

{

int temp = weight - a[i][end]; // определяем вес пути из предыдущей вершины

if (temp == d[i]) // если вес совпал с рассчитанным

{ // значит из этой вершины и был переход

weight = temp; // сохраняем новый вес

end = i; // сохраняем предыдущую вершину

ver[k] = i + 1; // и записываем ее в массив

k++;

}

}

}

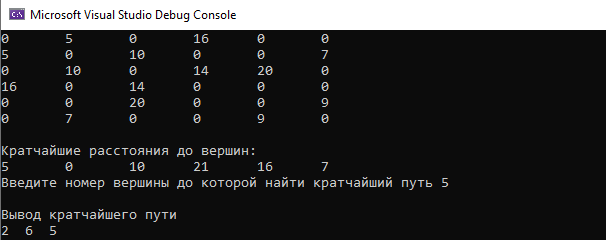
// Вывод пути (начальная вершина оказалась в конце массива из k элементов)

cout << "\nВывод кратчайшего пути" << endl;

for (int i = k - 1; i >= 0; i--)

cout << ver[i] << " ";

}

3. Результат работы****

## 

## **Лабораторная работа № 13**

## **STL – стандартная библиотека шаблонов в С++**

1. Постановка задачи

Написать и отладить три программы.

Первая программа демонстрирует использование контейнерных классов для хранения встроенных типов данных.

Вторая программа демонстрирует использование контейнерных классов для хранения пользовательских типов данных.

Третья программа демонстрирует использование алгоритмов STL.

**В программе №1** реализовать следующие задачи:

1. Создать объект-контейнер в соответствии с вариантом задания и заполнить его данными, тип которых определяется вариантом задания.

2. Просмотреть контейнер.

3. Изменить контейнер, удалив из него одни элементы и заменив другие.

4. Просмотреть контейнер, используя для доступа к его элементам итераторы.

5. Создать второй контейнер этого же класса и заполнить его данными того же типа, что и первый контейнер.

6. Изменить первый контейнер, удалив из него n элементов после заданного и добавив затем в него все элементы из второго контейнера.

7. Просмотреть первый и второй контейнеры.

**В программе № 2** выполнить то же самое, но для данных пользовательского типа.

**В программе № 3** реализовать следующие задачи:

1. Создать контейнер, содержащий объекты пользовательского типа. Тип контейнера выбирается в соответствии с вариантом задания.

2. Отсортировать его по убыванию элементов.

3. Просмотреть контейнер.

4. Используя подходящий алгоритм STL, найти в контейнере элемент, удовлетворяющий заданному условию, указанному в варианте.

5. Используя подходящий алгоритм STL переместить элементы, удовлетворяющие заданному условию в другой (предварительно пустой) контейнер. Тип второго контейнера определяется вариантом задания.

6. Просмотреть второй контейнер.

7. Используя подходящий алгоритм STL отсортировать первый и второй контейнеры по возрастанию элементов.

8. Просмотреть их.

9. Используя подходящий алгоритм STL получить третий контейнер путем слияния первых двух.

10. Просмотреть третий контейнер.

11. Используя подходящий алгоритм STL подсчитать, сколько элементов, удовлетворяющих заданному условию, содержит третий контейнер.

12. Используя подходящий алгоритм STL определить, есть ли в третьем контейнере элемент, удовлетворяющий заданному условию.

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | stack | queue | double |

1. Код программы

Task1.cpp

#include <iostream>

#include<iostream>

#include<stack>

#include<queue>

#include<ctime>

#include<vector>

using namespace std;

void fill\_stack(stack<double>& st, int n)

{

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

{

/\*cout << "Введите элемент: ";

double num;

cin >> num;\*/

double num = rand() % 50 - 10;

st.push(num);

}

}

void fill\_queue(queue<double>& q, int n)

{

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

{

/\*cout << "Введите элемент: ";

double num;

cin >> num;\*/

double num = rand() % 50 - 10;

q.push(num);

}

}

void print\_queue(queue<double>& q)

{

vector<double> vec;

while (!q.empty())

{

double num = q.front();

q.pop();

vec.push\_back(num);

cout << num << " ";

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

q.push(vec[i]);

}

}

void print\_stack(stack<double>& st)

{

vector<double> vec;

while (!st.empty())

{

double num = st.top();

st.pop();

vec.push\_back(num);

cout << num << " ";

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

//удаление всех отриц эл-тов эл-тов

void remove\_negative\_elements(stack<double>& st)

{

vector<double> vec;

while (!st.empty())

{

double num = st.top();

st.pop();

if (num > 0)

vec.push\_back(num);

}

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void replace\_elements(stack<double>& st)

{

vector<double> vec;

int i = 0;

int size = st.size();

while (!st.empty())

{

double num = st.top();

st.pop();

if (i == 0 || i == size - 1)

num = num \* 2;

vec.push\_back(num);

i++;

}

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void remove\_elements(stack<double>& st, double val, int count)

{

vector<double> vec;

bool find = false;

int i = 0;

while (!st.empty())

{

double num = st.top();

st.pop();

if (find && i < count)

{

i++;

continue;

}

vec.push\_back(num);

if (num == val)

find = true;

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void add\_stack(stack<double>& st1, queue<double>& q)

{

vector<double> vec;

while (!q.empty())

{

double num = q.front();

q.pop();

vec.push\_back(num);

st1.push(num);

}

for (int i = vec.size() - 1; i >= 0; i--)

{

q.push(vec[i]);

}

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

stack<double> st;

cout << "Кол-во эл-тов = ";

int n;

cin >> n;

fill\_stack(st, n);

cout << "Исходный стек" << endl;

print\_stack(st);

remove\_negative\_elements(st);

cout << "Стек после удаления отрицательных эл-тов" << endl;

print\_stack(st);

replace\_elements(st);

cout << "Стек после замены первого и последнего эл-том в два раза большим" << endl;

print\_stack(st);

queue<double> q;

cout << "Кол-во эл-тов = ";

cin >> n;

fill\_queue(q, n);

cout << "Второй контейнер (очередь)" << endl;

print\_queue(q);

double val;

cout << "Введите значение эл-та после которого нужно удалить элементы: ";

cin >> val;

int count;

cout << "Сколько эл-тов удалить? ";

cin >> count;

remove\_elements(st, val, count);

print\_stack(st);

add\_stack(st, q);

cout << "Первый стек после добавления второго контейнера" << endl;

print\_stack(st);

cout << "Второй контейнер" << endl;

print\_queue(q);

}

Task2.cpp

Book.h

#pragma once

#include <iostream>

using namespace std;

class book

{

private:

string name;

int cost;

public:

book(){}

book(string name, int cost)

{

this->name = name;

this->cost = cost;

}

string get\_name() { return name; }

void set\_name(string name) { this->name = name; }

int get\_cost() { return cost; }

void set\_cost(int cost) { this->cost = cost; }

//перегрузка оператора ввода (cin >> t)

friend istream& operator>>(istream& in, book& b);

//перегрузка оператора вывода (cout << t)

friend ostream& operator<<(ostream& out, const book& b);

};

Book.cpp

#include "book.h"

istream& operator>>(istream& in, book& b)

{

cout << "Название: ";

in >> b.name;

cout << "Цена: ";

in >> b.cost;

return in;

}

ostream& operator<<(ostream& out, const book& b)

{

out << "Название: " << b.name << "\tЦена: " << b.cost;

return out;

}

Main.cpp

#include <iostream>

#include<iostream>

#include<stack>

#include<queue>

#include<ctime>

#include<vector>

#include "book.h"

using namespace std;

void fill\_stack(stack<book\*>& st, int n)

{

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

{

book\* b = new book();

cin >> \*b;

st.push(b);

}

}

void fill\_queue(queue<book\*>& q, int n)

{

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

{

book\* b = new book();

cin >> \*b;

q.push(b);

}

}

void print\_stack(stack<book\*>& st)

{

vector<book\*> vec;

while (!st.empty())

{

book\* num = st.top();

st.pop();

vec.push\_back(num);

cout << \*num << endl;

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void print\_queue(queue<book\*>& q)

{

vector<book\*> vec;

while (!q.empty())

{

book\* num = q.front();

q.pop();

vec.push\_back(num);

cout << \*num << endl;

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

q.push(vec[i]);

}

}

//удаление всех книг стоимостью выше указанной

void remove\_books\_by\_cost(stack<book\*>& st, int cost)

{

vector<book\*> vec;

while (!st.empty())

{

book\* b = st.top();

st.pop();

if (b->get\_cost() < cost)

vec.push\_back(b);

}

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void replace\_first\_book(stack<book\*>& st, book\* new\_b)

{

vector<book\*> vec;

vec.push\_back(new\_b);

st.pop();

while (!st.empty())

{

book\* b = st.top();

st.pop();

vec.push\_back(b);

}

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void remove\_elements(stack<book\*>& st, string name, int count)

{

vector<book\*> vec;

bool find = false;

int i = 0;

while (!st.empty())

{

book\* num = st.top();

st.pop();

if (find && i < count)

{

i++;

continue;

}

vec.push\_back(num);

if (num->get\_name().compare(name) == 0)

find = true;

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void add\_stack(stack<book\*>& st1, queue<book\*>& q)

{

vector<book\*> vec;

while (!q.empty())

{

book\* num = q.front();

q.pop();

vec.push\_back(num);

st1.push(num);

}

for (int i = vec.size() - 1; i >= 0; i--)

{

q.push(vec[i]);

}

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

cout << "Создание стека книг" << endl;

stack<book\*> st;

cout << "Кол-во эл-тов = ";

int n;

cin >> n;

fill\_stack(st, n);

print\_stack(st);

cout << "Введите стоимость книги дороже которой нужно удалить: ";

int cost;

cin >> cost;

remove\_books\_by\_cost(st, cost);

print\_stack(st);

book\* new\_b = new book("X", 666);

replace\_first\_book(st, new\_b);

cout << "После замены первой книги на секретную" << endl;

print\_stack(st);

cout << "Создание очереди книг" << endl;

queue<book\*> q;

cout << "Кол-во эл-тов = ";

cin >> n;

fill\_queue(q, n);

cout << "Второй контейнер (очередь)" << endl;

print\_queue(q);

string name;

cout << "Введите название книги после которой нужно удалить элементы: ";

cin >> name;

int count;

cout << "Сколько эл-тов удалить? ";

cin >> count;

remove\_elements(st, name, count);

print\_stack(st);

add\_stack(st, q);

cout << "Первый стек после добавления второго контейнера" << endl;

print\_stack(st);

cout << "Второй контейнер" << endl;

print\_queue(q);

}

Task3.cpp

Book.h

#pragma once

#include <iostream>

using namespace std;

class book

{

private:

string name;

int cost;

public:

book() {}

book(string name, int cost)

{

this->name = name;

this->cost = cost;

}

string get\_name() { return name; }

void set\_name(string name) { this->name = name; }

int get\_cost() { return cost; }

void set\_cost(int cost) { this->cost = cost; }

//перегрузка оператора ввода (cin >> t)

friend istream& operator>>(istream& in, book& b);

//перегрузка оператора вывода (cout << t)

friend ostream& operator<<(ostream& out, const book& b);

bool operator>(const book& b);

bool operator<(const book& b);

};

Book.cpp

#include "book.h"

istream& operator>>(istream& in, book& b)

{

cout << "Название: ";

in >> b.name;

cout << "Цена: ";

in >> b.cost;

return in;

}

ostream& operator<<(ostream& out, const book& b)

{

out << "Название: " << b.name << "\tЦена: " << b.cost;

return out;

}

bool book::operator>(const book& b)

{

return cost > b.cost;

}

bool book::operator<(const book& b)

{

return cost < b.cost;

}

Main.cpp

#include <iostream>

#include<iostream>

#include<stack>

#include<queue>

#include<ctime>

#include<vector>

#include<algorithm>

#include "book.h"

using namespace std;

void fill\_stack(stack<book\*>& st, int n)

{

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

{

book\* b = new book();

cin >> \*b;

st.push(b);

}

}

void print\_stack(stack<book\*>& st)

{

vector<book\*> vec;

while (!st.empty())

{

book\* num = st.top();

st.pop();

vec.push\_back(num);

cout << \*num << endl;

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

bool compare\_book(book\* b1, book\* b2)

{

return \*b1 > \*b2;

}

void sort\_stack(stack<book\*>& st)

{

vector<book\*> vec;

while (!st.empty())

{

vec.push\_back(st.top());

st.pop();

}

sort(vec.begin(), vec.end(), compare\_book);

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

bool compare\_book\_down(book\* b1, book\* b2)

{

return \*b1 < \* b2;

}

void sort\_stack\_down(stack<book\*>& st)

{

vector<book\*> vec;

while (!st.empty())

{

vec.push\_back(st.top());

st.pop();

}

sort(vec.begin(), vec.end(), compare\_book\_down);

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

}

void sort\_queue\_down(queue<book\*>& st)

{

vector<book\*> vec;

while (!st.empty())

{

vec.push\_back(st.front());

st.pop();

}

sort(vec.begin(), vec.end(), compare\_book\_down);

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

st.push(vec[i]);

}

book\* find\_book(stack<book\*>& st, int cost)

{

vector<book\*> vec;

while (!st.empty())

{

vec.push\_back(st.top());

st.pop();

}

auto it = find\_if(vec.begin(), vec.end(), [cost](book\* b) { return b->get\_cost() > cost; });

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

if (it != vec.end())

return \*it;

return nullptr;

}

book\* find\_book(const vector<book\*>& vec, int cost, string name)

{

auto it = find\_if(vec.begin(), vec.end(), [cost, name](book\* b) { return b->get\_cost() == cost && b->get\_name().compare(name) == 0; });

if (it != vec.end())

return \*it;

return nullptr;

}

bool comp\_name\_start\_a(book\* b)

{

return tolower(b->get\_name()[0]) == 'a';

}

void move\_elements(stack<book\*>& st, queue<book\*>& q)

{

vector<book\*> vec;

while (!st.empty())

{

vec.push\_back(st.top());

st.pop();

}

vector<book\*> vec2(vec.size());

auto it = copy\_if(vec.begin(), vec.end(), vec2.begin(), comp\_name\_start\_a);

vec2.resize(distance(vec2.begin(), it));

auto new\_end = remove\_if(vec.begin(), vec.end(), comp\_name\_start\_a);

vec.erase(new\_end, vec.end());

for (int i = vec.size() - 1; i >= 0; i--)

{

st.push(vec[i]);

}

for (int i = 0; i < vec2.size(); i++)

{

q.push(vec2[i]);

}

}

void print\_queue(queue<book\*>& q)

{

vector<book\*> vec;

while (!q.empty())

{

book\* num = q.front();

q.pop();

vec.push\_back(num);

cout << \*num << endl;

}

cout << endl;

for (int i = vec.size() - 1; i >= 0; i--)

{

q.push(vec[i]);

}

}

vector<book\*> merge(stack<book\*>& st, queue<book\*>& q)

{

vector<book\*> vec1;

while (!st.empty())

{

vec1.push\_back(st.top());

st.pop();

}

vector<book\*> vec2;

while (!q.empty())

{

vec2.push\_back(q.front());

q.pop();

}

sort(vec1.begin(), vec1.end(), compare\_book\_down);

sort(vec2.begin(), vec2.end(), compare\_book\_down);

vector<book\*> vec3(vec1.size() + vec2.size());

merge(vec1.begin(), vec1.end(), vec2.begin(), vec2.end(), vec3.begin(), compare\_book\_down);

return vec3;

}

void print\_vector(const vector<book\*>& books)

{

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

{

cout << \*(books[i]) << endl;

}

cout << endl;

}

int count\_books(const vector<book\*>& books, char c)

{

int count = count\_if(books.begin(), books.end(), [c](book\* b) {return b->get\_name()[0] == c; });

return count;

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

cout << "Создание стека книг" << endl;

stack<book\*> st;

cout << "Кол-во эл-тов = ";

int n;

cin >> n;

fill\_stack(st, n);

print\_stack(st);

cout << "Сортировка по убыванию" << endl;

sort\_stack(st);

print\_stack(st);

int cost;

cout << "Поиск книги по стоимости" << endl;

cout << "Введите стоимость книги ";

cin >> cost;

book\* fbook = find\_book(st, cost);

if (fbook == nullptr)

cout << "Книга не найдена" << endl;

else

cout << \*fbook << endl;

queue<book\*> q;

cout << "Перемещение книг на букву А" << endl;

move\_elements(st, q);

cout << "Содержимое стека" << endl;

print\_stack(st);

cout << "Содержимое очереди" << endl;

print\_queue(q);

sort\_stack\_down(st);

sort\_queue\_down(q);

cout << "Сортировка по возрастанию стоимости" << endl;

cout << "Содержимое стека" << endl;

print\_stack(st);

cout << "Содержимое очереди" << endl;

print\_queue(q);

vector<book\*> vec = merge(st, q);

cout << "После слияния" << endl;

print\_vector(vec);

cout << "Подсчет кол-ва книг начинающихся на заданную букву" << endl;

cout << "Введите букву: ";

cin.ignore();

char c = getchar();

int count = count\_books(vec, c);

cout << "Кол-во книг = " << count << endl;

cout << "Поиск книги по названию и стоимости" << endl;

cout << "Введите стоимость книги ";

cin >> cost;

string name;

cout << "Введите название книги ";

cin >> name;

fbook = find\_book(vec, cost, name);

if (fbook == nullptr)

cout << "Книга не найдена" << endl;

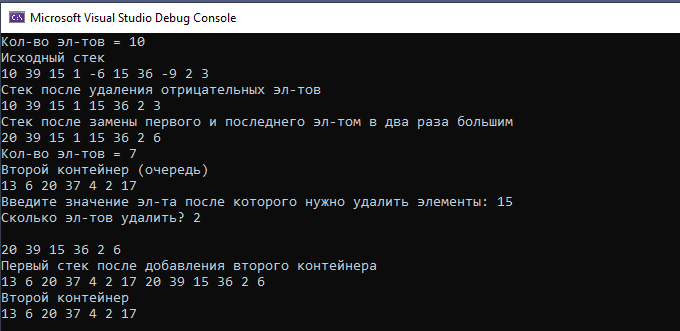
else

cout << \*fbook << endl;

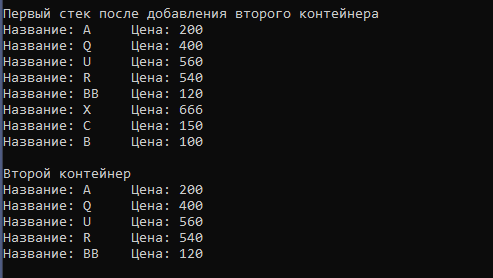
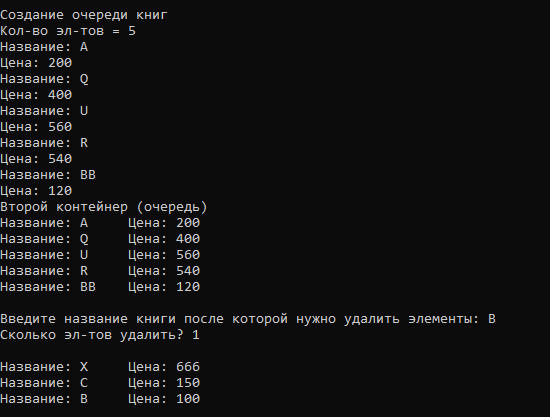
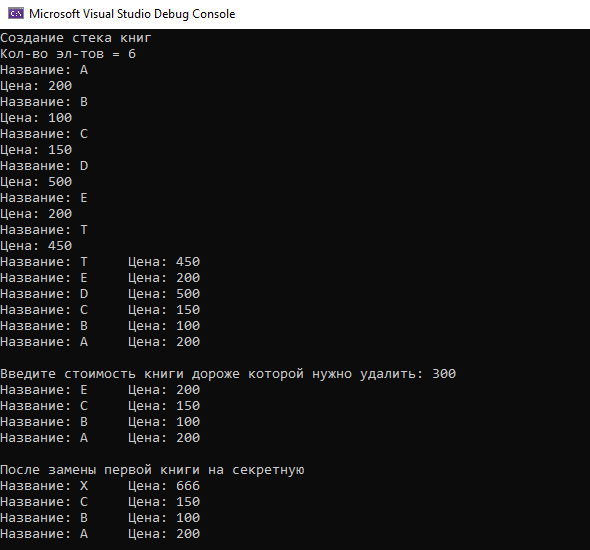
}

1. Результаты работы

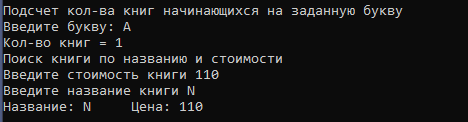
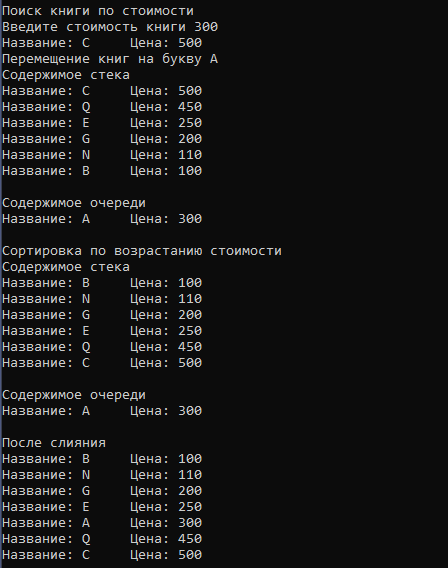
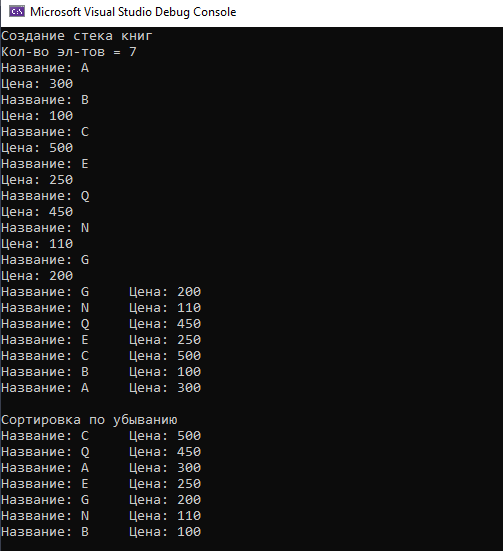
Программа 1



Программа 2



Программа 3



**Лабораторная работа № 14**

**Графическое приложение “Калькулятор”**

1.Постановка задания

Реализовать графическое приложение “Калькулятор”

2. Код программы

#pragma once

namespace Calculator {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

private:

double first;

String^ operation = "";

bool executeOperation = false;

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Button^ btnNum1;

private: System::Windows::Forms::Button^ btnNum2;

private: System::Windows::Forms::Button^ btnNum3;

private: System::Windows::Forms::Button^ btnNum6;

protected:

protected:

private: System::Windows::Forms::Button^ btnNum5;

private: System::Windows::Forms::Button^ btnNum4;

private: System::Windows::Forms::Button^ btnNum9;

private: System::Windows::Forms::Button^ btnNum8;

private: System::Windows::Forms::Button^ btnNum7;

private: System::Windows::Forms::Button^ btnNum0;

private: System::Windows::Forms::TextBox^ txtBxMain;

private: System::Windows::Forms::Button^ button11;

private: System::Windows::Forms::Button^ button12;

private: System::Windows::Forms::Button^ button13;

private: System::Windows::Forms::Button^ btnEqual;

private: System::Windows::Forms::Button^ btnDot;

private: System::Windows::Forms::Button^ button16;

private: System::Windows::Forms::Button^ button17;

private: System::Windows::Forms::Button^ btnRemoveChar;

private: System::Windows::Forms::Button^ btnClear;

private: System::Windows::Forms::Button^ btnSqrt;

private:

/// <summary>

/// Required designer variable.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->btnNum1 = (gcnew System::Windows::Forms::Button());

this->btnNum2 = (gcnew System::Windows::Forms::Button());

this->btnNum3 = (gcnew System::Windows::Forms::Button());

this->btnNum6 = (gcnew System::Windows::Forms::Button());

this->btnNum5 = (gcnew System::Windows::Forms::Button());

this->btnNum4 = (gcnew System::Windows::Forms::Button());

this->btnNum9 = (gcnew System::Windows::Forms::Button());

this->btnNum8 = (gcnew System::Windows::Forms::Button());

this->btnNum7 = (gcnew System::Windows::Forms::Button());

this->btnNum0 = (gcnew System::Windows::Forms::Button());

this->txtBxMain = (gcnew System::Windows::Forms::TextBox());

this->button11 = (gcnew System::Windows::Forms::Button());

this->button12 = (gcnew System::Windows::Forms::Button());

this->button13 = (gcnew System::Windows::Forms::Button());

this->btnEqual = (gcnew System::Windows::Forms::Button());

this->btnDot = (gcnew System::Windows::Forms::Button());

this->button16 = (gcnew System::Windows::Forms::Button());

this->button17 = (gcnew System::Windows::Forms::Button());

this->btnRemoveChar = (gcnew System::Windows::Forms::Button());

this->btnClear = (gcnew System::Windows::Forms::Button());

this->btnSqrt = (gcnew System::Windows::Forms::Button());

this->SuspendLayout();

//

// btnNum1

//

this->btnNum1->Location = System::Drawing::Point(41, 302);

this->btnNum1->Margin = System::Windows::Forms::Padding(4);

this->btnNum1->Name = L"btnNum1";

this->btnNum1->Size = System::Drawing::Size(59, 56);

this->btnNum1->TabIndex = 0;

this->btnNum1->Text = L"1";

this->btnNum1->UseVisualStyleBackColor = true;

this->btnNum1->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum2

//

this->btnNum2->Location = System::Drawing::Point(108, 302);

this->btnNum2->Margin = System::Windows::Forms::Padding(4);

this->btnNum2->Name = L"btnNum2";

this->btnNum2->Size = System::Drawing::Size(59, 56);

this->btnNum2->TabIndex = 1;

this->btnNum2->Text = L"2";

this->btnNum2->UseVisualStyleBackColor = true;

this->btnNum2->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum3

//

this->btnNum3->Location = System::Drawing::Point(175, 302);

this->btnNum3->Margin = System::Windows::Forms::Padding(4);

this->btnNum3->Name = L"btnNum3";

this->btnNum3->Size = System::Drawing::Size(59, 56);

this->btnNum3->TabIndex = 2;

this->btnNum3->Text = L"3";

this->btnNum3->UseVisualStyleBackColor = true;

this->btnNum3->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum6

//

this->btnNum6->Location = System::Drawing::Point(175, 238);

this->btnNum6->Margin = System::Windows::Forms::Padding(4);

this->btnNum6->Name = L"btnNum6";

this->btnNum6->Size = System::Drawing::Size(59, 56);

this->btnNum6->TabIndex = 5;

this->btnNum6->Text = L"6";

this->btnNum6->UseVisualStyleBackColor = true;

this->btnNum6->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum5

//

this->btnNum5->Location = System::Drawing::Point(108, 238);

this->btnNum5->Margin = System::Windows::Forms::Padding(4);

this->btnNum5->Name = L"btnNum5";

this->btnNum5->Size = System::Drawing::Size(59, 56);

this->btnNum5->TabIndex = 4;

this->btnNum5->Text = L"5";

this->btnNum5->UseVisualStyleBackColor = true;

this->btnNum5->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum4

//

this->btnNum4->Location = System::Drawing::Point(41, 238);

this->btnNum4->Margin = System::Windows::Forms::Padding(4);

this->btnNum4->Name = L"btnNum4";

this->btnNum4->Size = System::Drawing::Size(59, 56);

this->btnNum4->TabIndex = 3;

this->btnNum4->Text = L"4";

this->btnNum4->UseVisualStyleBackColor = true;

this->btnNum4->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum9

//

this->btnNum9->Location = System::Drawing::Point(175, 174);

this->btnNum9->Margin = System::Windows::Forms::Padding(4);

this->btnNum9->Name = L"btnNum9";

this->btnNum9->Size = System::Drawing::Size(59, 56);

this->btnNum9->TabIndex = 8;

this->btnNum9->Text = L"9";

this->btnNum9->UseVisualStyleBackColor = true;

this->btnNum9->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum8

//

this->btnNum8->Location = System::Drawing::Point(108, 174);

this->btnNum8->Margin = System::Windows::Forms::Padding(4);

this->btnNum8->Name = L"btnNum8";

this->btnNum8->Size = System::Drawing::Size(59, 56);

this->btnNum8->TabIndex = 7;

this->btnNum8->Text = L"8";

this->btnNum8->UseVisualStyleBackColor = true;

this->btnNum8->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum7

//

this->btnNum7->Location = System::Drawing::Point(41, 174);

this->btnNum7->Margin = System::Windows::Forms::Padding(4);

this->btnNum7->Name = L"btnNum7";

this->btnNum7->Size = System::Drawing::Size(59, 56);

this->btnNum7->TabIndex = 6;

this->btnNum7->Text = L"7";

this->btnNum7->UseVisualStyleBackColor = true;

this->btnNum7->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// btnNum0

//

this->btnNum0->Location = System::Drawing::Point(108, 367);

this->btnNum0->Margin = System::Windows::Forms::Padding(4);

this->btnNum0->Name = L"btnNum0";

this->btnNum0->Size = System::Drawing::Size(59, 56);

this->btnNum0->TabIndex = 9;

this->btnNum0->Text = L"0";

this->btnNum0->UseVisualStyleBackColor = true;

this->btnNum0->Click += gcnew System::EventHandler(this, &MyForm::btnNum1\_Click);

//

// txtBxMain

//

this->txtBxMain->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 18.25F));

this->txtBxMain->Location = System::Drawing::Point(41, 31);

this->txtBxMain->Name = L"txtBxMain";

this->txtBxMain->ReadOnly = true;

this->txtBxMain->Size = System::Drawing::Size(260, 35);

this->txtBxMain->TabIndex = 10;

this->txtBxMain->Text = L"0";

//

// button11

//

this->button11->Location = System::Drawing::Point(242, 174);

this->button11->Margin = System::Windows::Forms::Padding(4);

this->button11->Name = L"button11";

this->button11->Size = System::Drawing::Size(59, 56);

this->button11->TabIndex = 11;

this->button11->Text = L"Х";

this->button11->UseVisualStyleBackColor = true;

this->button11->Click += gcnew System::EventHandler(this, &MyForm::btnOperation\_Click);

//

// button12

//

this->button12->Location = System::Drawing::Point(242, 238);

this->button12->Margin = System::Windows::Forms::Padding(4);

this->button12->Name = L"button12";

this->button12->Size = System::Drawing::Size(59, 56);

this->button12->TabIndex = 12;

this->button12->Text = L"-";

this->button12->UseVisualStyleBackColor = true;

this->button12->Click += gcnew System::EventHandler(this, &MyForm::btnOperation\_Click);

//

// button13

//

this->button13->Location = System::Drawing::Point(242, 302);

this->button13->Margin = System::Windows::Forms::Padding(4);

this->button13->Name = L"button13";

this->button13->Size = System::Drawing::Size(59, 56);

this->button13->TabIndex = 13;

this->button13->Text = L"+";

this->button13->UseVisualStyleBackColor = true;

this->button13->Click += gcnew System::EventHandler(this, &MyForm::btnOperation\_Click);

//

// btnEqual

//

this->btnEqual->Location = System::Drawing::Point(242, 367);

this->btnEqual->Margin = System::Windows::Forms::Padding(4);

this->btnEqual->Name = L"btnEqual";

this->btnEqual->Size = System::Drawing::Size(59, 56);

this->btnEqual->TabIndex = 14;

this->btnEqual->Text = L"=";

this->btnEqual->UseVisualStyleBackColor = true;

this->btnEqual->Click += gcnew System::EventHandler(this, &MyForm::btnEqual\_Click);

//

// btnDot

//

this->btnDot->Location = System::Drawing::Point(175, 366);

this->btnDot->Margin = System::Windows::Forms::Padding(4);

this->btnDot->Name = L"btnDot";

this->btnDot->Size = System::Drawing::Size(59, 56);

this->btnDot->TabIndex = 15;

this->btnDot->Text = L",";

this->btnDot->UseVisualStyleBackColor = true;

this->btnDot->Click += gcnew System::EventHandler(this, &MyForm::btnDot\_Click);

//

// button16

//

this->button16->Location = System::Drawing::Point(41, 367);

this->button16->Margin = System::Windows::Forms::Padding(4);

this->button16->Name = L"button16";

this->button16->Size = System::Drawing::Size(59, 56);

this->button16->TabIndex = 16;

this->button16->Text = L"+/-";

this->button16->UseVisualStyleBackColor = true;

this->button16->Click += gcnew System::EventHandler(this, &MyForm::btnChangeSign\_Click);

//

// button17

//

this->button17->Location = System::Drawing::Point(242, 110);

this->button17->Margin = System::Windows::Forms::Padding(4);

this->button17->Name = L"button17";

this->button17->Size = System::Drawing::Size(59, 56);

this->button17->TabIndex = 17;

this->button17->Text = L"/";

this->button17->UseVisualStyleBackColor = true;

this->button17->Click += gcnew System::EventHandler(this, &MyForm::btnOperation\_Click);

//

// btnRemoveChar

//

this->btnRemoveChar->Location = System::Drawing::Point(108, 110);

this->btnRemoveChar->Margin = System::Windows::Forms::Padding(4);

this->btnRemoveChar->Name = L"btnRemoveChar";

this->btnRemoveChar->Size = System::Drawing::Size(59, 56);

this->btnRemoveChar->TabIndex = 18;

this->btnRemoveChar->Text = L"<-";

this->btnRemoveChar->UseVisualStyleBackColor = true;

this->btnRemoveChar->Click += gcnew System::EventHandler(this, &MyForm::btnRemoveChar\_Click);

//

// btnClear

//

this->btnClear->Location = System::Drawing::Point(41, 110);

this->btnClear->Margin = System::Windows::Forms::Padding(4);

this->btnClear->Name = L"btnClear";

this->btnClear->Size = System::Drawing::Size(59, 56);

this->btnClear->TabIndex = 19;

this->btnClear->Text = L"C";

this->btnClear->UseVisualStyleBackColor = true;

this->btnClear->Click += gcnew System::EventHandler(this, &MyForm::btnClear\_Click);

//

// btnSqrt

//

this->btnSqrt->Location = System::Drawing::Point(175, 110);

this->btnSqrt->Margin = System::Windows::Forms::Padding(4);

this->btnSqrt->Name = L"btnSqrt";

this->btnSqrt->Size = System::Drawing::Size(59, 56);

this->btnSqrt->TabIndex = 20;

this->btnSqrt->Text = L"Sqrt";

this->btnSqrt->UseVisualStyleBackColor = true;

this->btnSqrt->Click += gcnew System::EventHandler(this, &MyForm::btnSqrt\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(8, 17);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::Silver;

this->ClientSize = System::Drawing::Size(336, 458);

this->Controls->Add(this->btnSqrt);

this->Controls->Add(this->btnClear);

this->Controls->Add(this->btnRemoveChar);

this->Controls->Add(this->button17);

this->Controls->Add(this->button16);

this->Controls->Add(this->btnDot);

this->Controls->Add(this->btnEqual);

this->Controls->Add(this->button13);

this->Controls->Add(this->button12);

this->Controls->Add(this->button11);

this->Controls->Add(this->txtBxMain);

this->Controls->Add(this->btnNum0);

this->Controls->Add(this->btnNum9);

this->Controls->Add(this->btnNum8);

this->Controls->Add(this->btnNum7);

this->Controls->Add(this->btnNum6);

this->Controls->Add(this->btnNum5);

this->Controls->Add(this->btnNum4);

this->Controls->Add(this->btnNum3);

this->Controls->Add(this->btnNum2);

this->Controls->Add(this->btnNum1);

this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.25F));

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Margin = System::Windows::Forms::Padding(4);

this->MaximizeBox = false;

this->MinimizeBox = false;

this->Name = L"MyForm";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"Калькулятор";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void btnNum1\_Click(System::Object^ sender, System::EventArgs^ e) {

if (txtBxMain->Text == "0")

txtBxMain->Clear();

if (executeOperation)

{

txtBxMain->Clear();

executeOperation = false;

}

Button^ but = (Button^)sender;

txtBxMain->Text += but->Text;

}

private: System::Void EqualExecute()

{

if (operation == "")

return;

double second = Double::Parse(txtBxMain->Text);

double res = 0;

if (operation == "+")

res = first + second;

else if (operation == "-")

res = first - second;

else if (operation == "Х")

res = first \* second;

else if (operation == "/")

res = first / second;

first = res;

txtBxMain->Text = res.ToString();

operation = "";

}

private: System::Void btnOperation\_Click(System::Object^ sender, System::EventArgs^ e) {

if (txtBxMain->Text != "")

{

if (operation == "")

{

first = Double::Parse(txtBxMain->Text);

txtBxMain->Clear();

}

else

{

executeOperation = true;

EqualExecute();

}

Button^ but = (Button^)sender;

operation = but->Text;

}

}

private: System::Void btnEqual\_Click(System::Object^ sender, System::EventArgs^ e) {

EqualExecute();

}

private: System::Void btnChangeSign\_Click(System::Object^ sender, System::EventArgs^ e) {

double num = Double::Parse(txtBxMain->Text);

num = -num;

txtBxMain->Text = num.ToString();

}

private: System::Void btnClear\_Click(System::Object^ sender, System::EventArgs^ e) {

txtBxMain->Text = "0";

first = 0;

operation = "";

}

private: System::Void btnRemoveChar\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ s = txtBxMain->Text;

if (s->Length > 1)

s = s->Substring(0, s->Length - 1);

else

s = "0";

txtBxMain->Text = s;

}

private: System::Void btnSqrt\_Click(System::Object^ sender, System::EventArgs^ e) {

double num = Double::Parse(txtBxMain->Text);

if (num < 0)

{

MessageBox::Show("Введены неверные данные‬");

}

else

{

num = Math::Sqrt(num);

txtBxMain->Text = num.ToString();

}

}

private: System::Void btnDot\_Click(System::Object^ sender, System::EventArgs^ e) {

if (!txtBxMain->Text->Contains(","))

{

txtBxMain->Text += ",";

}

}

};

}

3. Результат работы

