Звіт

Автор: Чугунов В.Ю

КІТ-119а

Лабораторна робота №13

**STL. АЛГОРИТМИ ПЕРЕМІЩЕННЯ ТА ПОШУКУ**

**Мета:** На практиці порівняти STL алгоритми, що не модифікують послідовність.

1 ЗАВДАННЯ ДО РОБОТИ

**Загальне завдання.** Поширити попередню лабораторну роботу, додаючи наступні можливості діалогового меню:

* вивід всіх елементів масиву за допомогою STL функції f*or\_each*;
* визначення кількості елементів за заданим критерієм (рис. 2);
* пошук елементу за заданим критерієм (рис. 3).

2 РОЗРОБКА АЛГОРИТМІВ ФУНКЦІЙ

**Клас ConsoleUI<typename T>**

**private поля:**

vector<T> Vector – контейнер std::vector<>, для використання: #include <vector>

set<T> Set – контейнер std::set<>, для використання: #include <set>

list<T> List – контейнер std::list<>, для використання: #include <list>

map<int, T> Map – контейнер std::map<>, для використання #include <map>

**Функції:  
 int main ( )** Призначення: головна функція.  
 Опис роботи:   
 1) створення об’єкта класу ConsoleUI;   
 2) виклик метода ConsoleUI::RunExe.

**void RunExe ( )** Призначення: головне меню програми.  
 Опис роботи: користувачу пропонується вибрати один з контейнерів. Будь-який з варіантів призведе до виклику вибору функцій, що відповідають за обраний контейнер. Функція буде працювати до тих пір, поки користувач не введе «0» (умова виходу з програми).

**void VectorCount ( int ch )**  
 Призначення: рахування елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв рахування елементів вектору («0» - скасування операції). Після вибору виконується рахування, виведення результату на екран.

**void ListCount ( int ch )** Призначення: рахування елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв рахування елементів вектору («0» - скасування операції). Після вибору виконується рахування, виведення результату на екран.

**void SetCount ( int ch )** Призначення: рахування елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв рахування елементів вектору («0» - скасування операції). Після вибору виконується рахування, виведення результату на екран.

**void MapCount ( int ch )** Призначення: рахування елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв рахування елементів вектору («0» - скасування операції). Після вибору виконується рахування, виведення результату на екран.

**void VectorFind ( int ch )**  
 Призначення: пошук елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв пошуку елементів вектору («0» - скасування операції). Після вибору виконується пошук елементів, виведення результату на екран.

**void ListFind ( int ch )**  
 Призначення: пошук елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв пошуку елементів вектору («0» - скасування операції). Після вибору виконується пошук елементів, виведення результату на екран.

**void SetFind ( int ch )**  
 Призначення: пошук елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв пошуку елементів вектору («0» - скасування операції). Після вибору виконується пошук елементів, виведення результату на екран.

**void MapFind ( int ch )**  
 Призначення: пошук елементів за заданим критерієм.  
 Опис роботи: перевіряється наявність елементів у контейнері. Якщо елементів не має, то функція виводить на екран відповідне повідомлення. В іншому випадку користувачу пропонується обрати один з трьох критеріїв пошуку елементів вектору («0» - скасування операції). Після вибору виконується пошук елементів, виведення результату на екран.

3 ТЕКСТ ПРОГРАМИ

**Текст файлу main.cpp**

#include "Dialog.h"

int main()

{

setlocale(LC\_ALL, "rus");

ConsoleUI<Point> UI;

UI.RunExe();

system("PAUSE");

return 0;

}

**Текст файлу Dialog.h**

#pragma once

#include <vector>

#include <set>

#include <list>

#include <map>

#include <algorithm>

#include <iterator>

#include <Windows.h>

#include"Point.h"

void SetColor(int text, int background)

{

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));

}

using std::vector;

using std::set;

using std::list;

using std::map;

using std::for\_each;

using std::count\_if;

using std::copy\_if;

using std::inserter;

using std::back\_inserter;

using std::cout;

using std::cin;

using std::endl;

template<typename T>

class ConsoleUI

{

public:

/\* Меню - отвечает за работу диалогового окна \*/

void RunExe();

private:

// Используемые контейнеры

vector<T> Vector;

set<T> Set;

list<T> List;

map<int, T> Map;

// Выбор контейнера

int ContainerSelect();

// Выбор операции

int OperationSelect();

// Выполнение операции над Вектором

void DoVectorOperation(int ch);

// Добавление элемента в Вектор

void AddToVector(int ch);

// Удаление элемента из вектора

void DelFromVector(int ch);

// Получение данных из вектора

T VectorGet(int ch);

// Вывод содержимого Вектора на экран

void VectorShow();

// Подсчет элементов по заданому критерию

void VectorCount(int ch);

// Поиск элементов по заданому критерию

void VectorFind(int ch);

// Уведомление об отсутствии данных в Векторе

bool NoVectorData();

// Уведомление о выборе Вектора

void ChoseVector();

// Выполнение операции над Сетом

void DoSetOperation(int ch);

// Добавление элемента в Сет

void AddToSet();

// Удаление элемента из Сета

void DelFromSet();

// Получение данных из Сета

T SetGet();

// Вывод содержимого Сета на экран

void SetShow();

// Подсчет элементов по заданому критерию

void SetCount(int ch);

// Поиск элементов по заданому критерию

void SetFind(int ch);

// Уведомление об отсутствии данных в Сете

bool NoSetData();

// Уведомление о выборе Сета

void ChoseSet();

// Выполнение операции над Списком

void DoListOperation(int ch);

// Добавление элемента в Список

void AddToList(int ch);

// Удаление элемента из Списка

void DelFromList(int ch);

// Получение данных из Списка

T ListGet(int ch);

// Вывод содержимого Списка на экран

void ListShow();

// Подсчет элементов по заданому критерию

void ListCount(int ch);

// Поиск элементов по заданому критерию

void ListFind(int ch);

// Уведомление об отсутствии данных в Списке

bool NoListData();

// Уведомление о выборе Списка

void ChoseList();

// Выпонение операции над Мапом

void DoMapOperation(int ch);

// Добавление элемента в Мап

void AddToMap();

// Удаление элемента из Мапа

void DelFromMap();

// Получение данных из Мапа

T MapGet();

// Вывод содержимого Мапа на экран

void MapShow();

// Подсчет элементов по заданому критерию

void MapCount(int ch);

// Поиск элементов по заданому критерию

void MapFind(int ch);

// Уведомление об отсутствии данных в Мапе

bool NoMapData();

// Уведомление о выборе Мапа

void ChoseMap();

// Уведомление об успешной операции

void OperationPass();

// Уведомление об отмене операции

void OperationCancel();

// Информация о лабораторной работе

void LabInfo();

// Проверка выбора варианта

bool CheckSelect(int b, int e, int i);

};

// ########################## ГЛАВНОЕ МЕНЮ ##########################

template<typename T>

inline void ConsoleUI<T>::RunExe()

{

int choice;

do {

back:

choice = ContainerSelect();

switch (choice)

{

case 1:

DoVectorOperation(choice);

system("cls");

break;

case 2:

DoListOperation(choice);

system("cls");

break;

case 3:

DoSetOperation(choice);

system("cls");

break;

case 4:

DoMapOperation(choice);

system("cls");

break;

case 0:

SetColor(13, 0);

cout << "\nСпасибо за работу!\n";

SetColor(7, 0);

}

} while (choice);

}

// ######################## ВЫБОР КОНТЕЙНЕРА ########################

template<typename T>

inline int ConsoleUI<T>::ContainerSelect()

{

int choice;

LabInfo();

cout << "Выберите контейнер:\n";

cout << "1. Vector.\n";

cout << "2. List.\n";

cout << "3. Set.\n";

cout << "4. Map.\n";

cout << "0. Выход.\n";

cout << "Введите ваш ответ сюда: ";

cin >> choice;

CheckSelect(0, 4, choice);

return choice;

}

// ######################### ВЫБОР ОПЕРАЦИЙ #########################

template<typename T>

inline int ConsoleUI<T>::OperationSelect()

{

int choice;

cout << "Список доступных операций:\n";

cout << "1. Добавление элемента.\n";

cout << "2. Удаление элемента.\n";

cout << "3. Получение данных.\n";

cout << "4. Вывод содержимого на экран.\n";

cout << "5. Подсчет количества элементов.\n";

cout << "6. Поиск элементов.\n";

cout << "0. Вернуться к выбору контейнера.\n";

cout << "Введите ваш ответ сюда: ";

cin >> choice;

CheckSelect(0, 6, choice);

return choice;

}

// ######################## ОПЕРАЦИЯ УСПЕШНА ########################

template<typename T>

inline void ConsoleUI<T>::OperationPass()

{

SetColor(10, 0);

cout << "Операция прошла успешно!\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

}

// ######################### ОТМЕНА ОПЕРАЦИИ ########################

template<typename T>

inline void ConsoleUI<T>::OperationCancel()

{

SetColor(12, 0);

cout << "Операция отменена.\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

}

// ########################### ИНФО О ЛАБЕ ##########################

template<typename T>

inline void ConsoleUI<T>::LabInfo()

{

SetColor(9, 0);

cout << " Лабораторная работа №12: \"STL контейнеры\" \*Выполнил: Пумня Александр\n";

SetColor(7, 0);

}

// ################## ПРОВЕРКА ПРАВИЛЬНОСТИ ВЫБОРА ##################

template<typename T>

inline bool ConsoleUI<T>::CheckSelect(int b, int e, int i)

{

if (i < b || i > e)

{

SetColor(12, 0);

cout << "\nВы ошиблись. Введите число от " << b << " до " << e << endl;

SetColor(7, 0);

system("PAUSE");

system("cls");

return false;

}

return true;

}

template<typename T> // Предикатор вывода

void Print\_Pred(const T &elem)

{

cout << elem << " ";

}

template<typename T> // Предикатор вывода map

void Print\_Map(std::pair<int, T> elem)

{

cout << '[' << elem.first << "] : " << elem.second << endl;

}

#include "Vector.hpp" // Реализация функций над Вектором

#include "Set.hpp" // Реализация функций над Сетом

#include "List.hpp" // Реализация функций над Списком

#include "Map.hpp" // Реализация функций над Мапом

**Текст файлу Vector.hpp**

#pragma once

#include "Dialog.h"

// ##################### ОПЕРАЦИИ НАД ВЕКТОРОМ ######################

template<typename T>

inline void ConsoleUI<T>::DoVectorOperation(int ch)

{

do {

ChoseVector();

ch = OperationSelect();

switch (ch)

{

case 1:

AddToVector(ch);

break;

case 2:

DelFromVector(ch);

break;

case 3:

VectorGet(ch);

break;

case 4:

VectorShow();

break;

case 5:

VectorCount(ch);

break;

case 6:

VectorFind(ch);

break;

case 0:

return;

}

} while (ch);

}

// ###################### ДАБАВЛЕНИЕ В ВЕКТОР #######################

template<typename T>

inline void ConsoleUI<T>::AddToVector(int ch)

{

T value;

int index;

auto iterator = Vector.begin();

do

{

ChoseVector();

cout << "Выберите способ добавления:\n";

cout << "1. Добавление в конец.\n";

cout << "2. Вставка.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 2, ch));

switch (ch)

{

case 1:

cout << "\nВведите значение для добавления: ";

cin >> value;

Vector.push\_back(value);

OperationPass();

break;

case 2:

cout << "\nВведите значение для добавления: ";

cin >> value;

do

{

cout << "Введите индекс: ";

cin >> index;

if (index > Vector.size() || index < 0)

{

SetColor(12, 0);

cout << "Вы ошиблись. Введите число от 0 до " << Vector.size() << endl;

SetColor(7, 0);

}

} while (index > Vector.size() || index < 0);

std::advance(iterator, index);

Vector.insert(iterator, value);

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ###################### УДАЛЕНИЕ ИЗ ВЕКТОРА #######################

template<typename T>

inline void ConsoleUI<T>::DelFromVector(int ch)

{

if (NoVectorData())

return;

int index;

auto iterator = Vector.begin();

do

{

ChoseVector();

cout << "Выберите способ удаления:\n";

cout << "1. Удаление с конца.\n";

cout << "2. Удаление по индексу.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 2, ch));

switch (ch)

{

case 1:

Vector.pop\_back();

cout << endl;

OperationPass();

break;

case 2:

do

{

cout << "Введите индекс: ";

cin >> index;

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

{

SetColor(12, 0);

cout << "Вы ошиблись. Введите число от 0 до " << Vector.size() - 1 << endl;

SetColor(7, 0);

}

} while (index > Vector.size() || index < 0);

std::advance(iterator, index);

Vector.erase(iterator);

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ################## ПОЛУЧЕНИЕ ДАННЫХ ИЗ ВЕКТОРА ###################

template<typename T>

inline T ConsoleUI<T>::VectorGet(int ch)

{

if (NoVectorData())

return T();

int index;

auto iterator = Vector.begin();

do

{

ChoseVector();

cout << "Выберите вариант получения:\n";

cout << "1. Вывести первый элемент.\n";

cout << "2. Вывести элемент по индексу.\n";

cout << "3. Вывести последний элемент.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << endl << Vector.front() << endl;

OperationPass();

return Vector.front();

break;

case 2:

do

{

cout << "Введите индекс: ";

cin >> index;

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

{

SetColor(12, 0);

cout << "Вы ошиблись. Введите число от 0 до " << Vector.size() - 1 << endl;

SetColor(7, 0);

}

} while (index > Vector.size() || index < 0);

cout << endl << Vector[index] << endl;

OperationPass();

return Vector[index];

break;

case 3:

cout << endl << Vector.back() << endl;

OperationPass();

return Vector.back();

break;

case 0:

cout << endl;

OperationCancel();

}

return T();

}

// ####################### ПОДСЧЕТ ЭЛЕМЕНТОВ ########################

template<typename T>

inline void ConsoleUI<T>::VectorCount(int ch)

{

if (NoVectorData())

return;

T value;

int count;

do

{

ChoseVector();

cout << "Выберите критерий подсчета:\n";

cout << "1. Посчитать все элементы, которые больше заданого.\n";

cout << "2. Посчитать все элементы, которые меньше заданого.\n";

cout << "3. Посчитать все элементы, которые равны заданому.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(Vector.begin(), Vector.end(), [&value](const T &elem) { return value < elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 2:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(Vector.begin(), Vector.end(), [&value](const T &elem) { return value > elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 3:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(Vector.begin(), Vector.end(), [&value](const T &elem) { return value == elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ######################## ПОИСК ЭЛЕМЕНТОВ #########################

template<typename T>

inline void ConsoleUI<T>::VectorFind(int ch)

{

if (NoVectorData())

return;

T value;

vector<T> temp;

do

{

ChoseVector();

cout << "Выберите критерий подсчета:\n";

cout << "1. Найти все элементы, которые больше заданого.\n";

cout << "2. Найти все элементы, которые меньше заданого.\n";

cout << "3. Найти все элементы, которые равны заданому.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(Vector.begin(), Vector.end(), back\_inserter(temp) , [&value](const T &elem) { return value < elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 2:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(Vector.begin(), Vector.end(), back\_inserter(temp), [&value](const T &elem) { return value > elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 3:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(Vector.begin(), Vector.end(), back\_inserter(temp), [&value](const T &elem) { return value == elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ######################### ВЫВОД ВЕКТОРА ##########################

template<typename T>

inline void ConsoleUI<T>::VectorShow()

{

if (NoVectorData())

return;

cout << endl;

for\_each(Vector.begin(), Vector.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

}

// ####################### НЕТ ДАННЫХ ВЕКТОРА #######################

template<typename T>

inline bool ConsoleUI<T>::NoVectorData()

{

if (!Vector.size())

{

SetColor(12, 0);

cout << "\nДанные отсутствуют!\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

return true;

}

return false;

}

// ######################### ВЫБОР ВЕКТОРА ##########################

template<typename T>

inline void ConsoleUI<T>::ChoseVector()

{

system("cls");

LabInfo();

cout << "Выбран контейнер: ";

SetColor(11, 0);

cout << "vector\n";

SetColor(7, 0);

}

**Текст файлу Set.hpp**

#pragma once

#include "Dialog.h"

// ####################### ОПЕРАЦИИ НАД СЕТОМ #######################

template<typename T>

inline void ConsoleUI<T>::DoSetOperation(int ch)

{

do {

ChoseSet();

ch = OperationSelect();

switch (ch)

{

case 1:

AddToSet();

break;

case 2:

DelFromSet();

break;

case 3:

SetGet();

break;

case 4:

SetShow();

break;

case 5:

SetCount(ch);

break;

case 6:

SetFind(ch);

break;

case 0:

return;

}

} while (ch);

}

// ######################## ДОБАВЛЕНИЕ В СЕТ ########################

template<typename T>

inline void ConsoleUI<T>::AddToSet()

{

T value;

cout << "\nВведите данные для добавления: ";

cin >> value;

Set.insert(value);

OperationPass();

}

// ######################## УДАЛЕНИЕ ИЗ СЕТА ########################

template<typename T>

inline void ConsoleUI<T>::DelFromSet()

{

if (NoSetData())

return;

T value;

cout << "\nВведите удаляемые данные: ";

cin >> value;

auto iterator = Set.find(value);

if (iterator == Set.end())

{

SetColor(12, 0);

cout << "Данных с таким значением нет в контейнере.\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

return;

}

Set.erase(iterator);

OperationPass();

}

// #################### ПОЛУЧЕНИЕ ДАННЫХ ИЗ СЕТА ####################

template<typename T>

inline T ConsoleUI<T>::SetGet()

{

if (NoSetData())

return T();

T value;

cout << "\nВведите искомые данные: ";

cin >> value;

auto iterator = Set.find(value);

if (iterator == Set.end())

{

SetColor(12, 0);

cout << "Данных с таким значением нет в контейнере.\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

return T();

}

cout << "Данные с таким значением есть в контейнере и успешно возвращены!\n";

OperationPass();

return \*iterator;

}

// ####################### ПОДСЧЕТ ЭЛЕМЕНТОВ ########################

template<typename T>

inline void ConsoleUI<T>::SetCount(int ch)

{

if (NoSetData())

return;

T value;

int count;

do

{

ChoseSet();

cout << "Выберите критерий подсчета:\n";

cout << "1. Посчитать все элементы, которые больше заданого.\n";

cout << "2. Посчитать все элементы, которые меньше заданого.\n";

cout << "3. Посчитать все элементы, которые равны заданому.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(Set.begin(), Set.end(), [&value](const T &elem) { return value < elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 2:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(Set.begin(), Set.end(), [&value](const T &elem) { return value > elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 3:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(Set.begin(), Set.end(), [&value](const T &elem) { return value == elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ######################## ПОИСК ЭЛЕМЕНТОВ #########################

template<typename T>

inline void ConsoleUI<T>::SetFind(int ch)

{

if (NoSetData())

return;

T value;

set<T> temp;

do

{

ChoseSet();

cout << "Выберите критерий подсчета:\n";

cout << "1. Найти все элементы, которые больше заданого.\n";

cout << "2. Найти все элементы, которые меньше заданого.\n";

cout << "3. Найти все элементы, которые равны заданому.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(Set.begin(), Set.end(), inserter(temp, temp.end()), [&value](const T &elem) { return value < elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 2:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(Set.begin(), Set.end(), inserter(temp, temp.end()), [&value](const T &elem) { return value > elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 3:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(Set.begin(), Set.end(), inserter(temp, temp.end()), [&value](const T &elem) { return value == elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ########################### ВЫВОД СЕТА ###########################

template<typename T>

inline void ConsoleUI<T>::SetShow()

{

if (NoSetData())

return;

cout << endl;

for\_each(Set.begin(), Set.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

}

// ######################## НЕТ ДАННЫХ СЕТА #########################

template<typename T>

inline bool ConsoleUI<T>::NoSetData()

{

if (!Set.size())

{

SetColor(12, 0);

cout << "\nДанные отсутствуют!\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

return true;

}

return false;

}

// ########################### ВЫБОР СЕТА ###########################

template<typename T>

inline void ConsoleUI<T>::ChoseSet()

{

system("cls");

LabInfo();

cout << "Выбран контейнер: ";

SetColor(11, 0);

cout << "set\n";

SetColor(7, 0);

}

**Текст файлу List.hpp**

#pragma once

#include "Dialog.h"

// ###################### ОПЕРАЦИИ НАД СПИСКОМ ######################

template<typename T>

inline void ConsoleUI<T>::DoListOperation(int ch)

{

do {

ChoseList();

ch = OperationSelect();

switch (ch)

{

case 1:

AddToList(ch);

break;

case 2:

DelFromList(ch);

break;

case 3:

ListGet(ch);

break;

case 4:

ListShow();

break;

case 5:

ListCount(ch);

break;

case 6:

ListFind(ch);

break;

case 0:

return;

}

} while (ch);

}

// ###################### ДОБАВЛЕНИЕ В СПИСОЕ #######################

template<typename T>

inline void ConsoleUI<T>::AddToList(int ch)

{

T value;

int index;

auto iterator = List.begin();

do

{

ChoseList();

cout << "Выберите способ добавления:\n";

cout << "1. Добавление в начало.\n";

cout << "2. Вставка.\n";

cout << "3. Добавление в конец.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nВведите значение для добавления: ";

cin >> value;

List.push\_front(value);

OperationPass();

break;

case 2:

cout << "\nВведите значение для добавления: ";

cin >> value;

do

{

cout << "Введите индекс: ";

cin >> index;

if (index > List.size() || index < 0)

{

SetColor(12, 0);

cout << "Вы ошиблись. Введите число от 0 до " << List.size() << endl;

SetColor(7, 0);

}

} while (index > List.size() || index < 0);

std::advance(iterator, index);

List.insert(iterator, value);

OperationPass();

break;

case 3:

cout << "\nВведите значение для добавления: ";

cin >> value;

List.push\_back(value);

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ####################### УДАЛЕНИЕ ИЗ СПИСКА #######################

template<typename T>

inline void ConsoleUI<T>::DelFromList(int ch)

{

if (NoListData())

return;

int index;

auto iterator = List.begin();

do

{

ChoseList();

cout << "Выберите способ удаления:\n";

cout << "1. Удаление с начала.\n";

cout << "2. Удаление по индексу.\n";

cout << "3. Удаление с конца.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

List.pop\_front();

cout << endl;

OperationPass();

break;

case 2:

do

{

cout << "Введите индекс: ";

cin >> index;

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

{

SetColor(12, 0);

cout << "Вы ошиблись. Введите число от 0 до " << List.size() - 1 << endl;

SetColor(7, 0);

}

} while (index > List.size() || index < 0);

std::advance(iterator, index);

List.erase(iterator);

OperationPass();

break;

case 3:

List.pop\_back();

cout << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ################### ПОЛУЧЕНИЕ ДАННЫХ ИЗ СПИСКА ###################

template<typename T>

inline T ConsoleUI<T>::ListGet(int ch)

{

if (NoListData())

return T();

int index;

auto iterator = List.begin();

do

{

ChoseList();

cout << "Выберите вариант получения:\n";

cout << "1. Вывести первый элемент.\n";

cout << "2. Вывести элемент по индексу.\n";

cout << "3. Вывести последний элемент.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << endl << List.front() << endl;

OperationPass();

return List.front();

break;

case 2:

do

{

cout << "Введите индекс: ";

cin >> index;

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

{

SetColor(12, 0);

cout << "Вы ошиблись. Введите число от 0 до " << List.size() - 1 << endl;

SetColor(7, 0);

}

} while (index > List.size() || index < 0);

advance(iterator, index);

cout << endl << \*iterator << endl;

OperationPass();

return \*iterator;

break;

case 3:

cout << endl << List.back() << endl;

OperationPass();

return List.back();

break;

case 0:

cout << endl;

OperationCancel();

}

return T();

}

// ####################### ПОДСЧЕТ ЭЛЕМЕНТОВ ########################

template<typename T>

inline void ConsoleUI<T>::ListCount(int ch)

{

if (NoListData())

return;

T value;

int count;

do

{

ChoseList();

cout << "Выберите критерий подсчета:\n";

cout << "1. Посчитать все элементы, которые больше заданого.\n";

cout << "2. Посчитать все элементы, которые меньше заданого.\n";

cout << "3. Посчитать все элементы, которые равны заданому.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(List.begin(), List.end(), [&value](const T &elem) { return value < elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 2:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(List.begin(), List.end(), [&value](const T &elem) { return value > elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 3:

cout << "\nЗадайте значение: ";

cin >> value;

count = count\_if(List.begin(), List.end(), [&value](const T &elem) { return value == elem; });

cout << "\n\nКоличество элементов, подходящих по критерию: " << count << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ######################## ПОИСК ЭЛЕМЕНТОВ #########################

template<typename T>

inline void ConsoleUI<T>::ListFind(int ch)

{

if (NoListData())

return;

T value;

list<T> temp;

do

{

ChoseList();

cout << "Выберите критерий подсчета:\n";

cout << "1. Найти все элементы, которые больше заданого.\n";

cout << "2. Найти все элементы, которые меньше заданого.\n";

cout << "3. Найти все элементы, которые равны заданому.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

} while (!CheckSelect(0, 3, ch));

switch (ch)

{

case 1:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(List.begin(), List.end(), back\_inserter(temp), [&value](const T &elem) { return value < elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 2:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(List.begin(), List.end(), back\_inserter(temp), [&value](const T &elem) { return value > elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 3:

cout << "\nЗадайте значение: ";

cin >> value;

copy\_if(List.begin(), List.end(), back\_inserter(temp), [&value](const T &elem) { return value == elem; });

cout << "\n\nСписок элементов, подходящих по критерию:\n";

for\_each(temp.begin(), temp.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

// ########################## ВЫВОД СПИСКА ##########################

template<typename T>

inline void ConsoleUI<T>::ListShow()

{

if (NoListData())

return;

cout << endl;

for\_each(List.begin(), List.end(), Print\_Pred<T>);

cout << endl;

OperationPass();

}

// ####################### НЕТ ДАННЫХ СПИСКА ########################

template<typename T>

inline bool ConsoleUI<T>::NoListData()

{

if (!List.size())

{

SetColor(12, 0);

cout << "\nДанные отсутствуют!\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

return true;

}

return false;

}

// ########################## ВЫБОР СПИСКА ##########################

template<typename T>

inline void ConsoleUI<T>::ChoseList()

{

system("cls");

LabInfo();

cout << "Выбран контейнер: ";

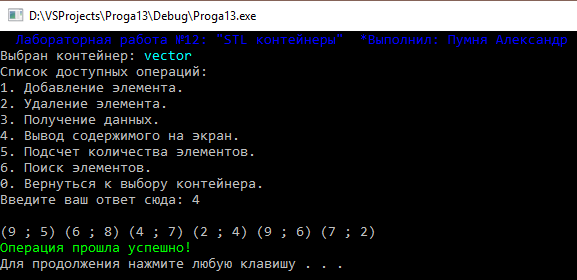
SetColor(11, 0);

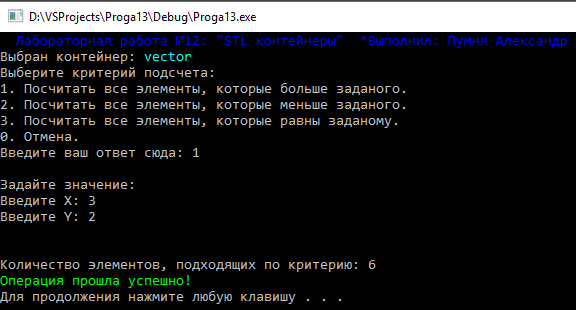
cout << "list\n";

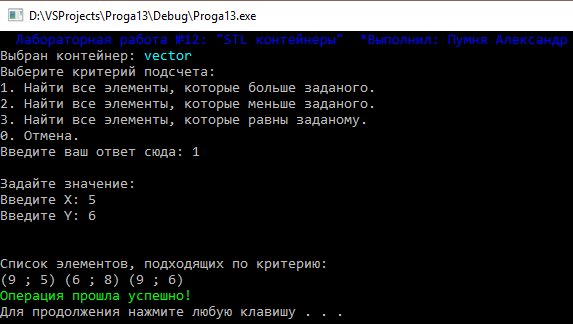
SetColor(7, 0);

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

  
Рисунок 1 – Елементи вектору

  
Рисунок 2 – Результат розрахунку кількості елементів

  
Рисунок 3 – Результат пошуку елементів

ВИСНОВКИ

При виконанні лабораторної роботи було реалізовано діалогове меню для вибору критерію розрахунку кількості елементів колекції та діалогове меню для вибору критерію пошуку елементів колекції. Досліджено алгоритми, що не модифікують колекції.