Звіт

Автор: Чугунов В.Ю

КІТ-119а

Лабораторна робота №14

**STL. АЛГОРИТМИ ЗМІНИ ПОСЛІДОВНОСТІ. СОРТУВАННЯ. ФУНКТОРИ**

**Мета:** На практиці порівняти STL алгоритми, що модифікують послідовність. Отримати навички роботи з STL функторами.

1 ЗАВДАННЯ ДО РОБОТИ

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

* об’єднання двох контейнерів типу vector;
* сортувати заданий контейнер з використанням функтора.

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.  
 Схема алгоритму подана на рис. 1.

**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).  
 Схема алгоритму подана на рис. 2.

**void VectorMerge ( )**  
 Призначення: злиття двох векторів типу *vector*.  
 Опис роботи: функція створює ще один вектор, пропонує його заповнити значеннями ( «-» - для виходу). Потім використовує функцію -std::copy для того, щоб об’єднати вже існуючий вектор з новим.

**void VectorSort ( int ch )**  
 Призначення: сортування вектору за заданим критерієм.  
 Опис роботи: функція пропонує обрати спосіб сортування («0» - вихід). Сортувати можна за зростанням та за спаданням. Після вибору виконається сортування вектору.

**Схеми алгоритмів функцій.**

Рисунок 1 – Схема алгоритму функції main

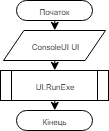
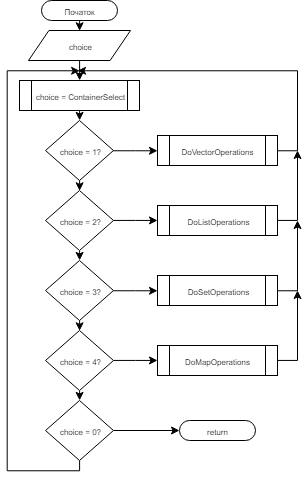


Рисунок 2 – Схема алгоритму функції RunExe



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::sort;

using std::cout;

using std::cin;

using std::endl;

template<typename T>

class SortingMinMax

{

public:

bool operator() (const T &a, const T &b) { return a < b; };

};

template<typename T>

class SortingMaxMin

{

public:

bool operator() (const T &a, const T &b) { return a > b; };

};

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);

// Слияние Векторов

void VectorMerge();

// Сортировка Вектора по заданому критерию

void VectorSort(int ch);

// Уведомление об отсутствии данных в Векторе

bool NoVectorData();

// Уведомление о выборе Вектора

void ChoseVector();

// 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);

// Сортировка списка по заданому критерию

void ListSort(int ch);

// Уведомление об отсутствии данных в Списке

bool NoListData();

// Уведомление о выборе Списка

void ChoseList();

// 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();

// 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 << "7. Слияние контейнеров.\n";

cout << "8. Сортировка по критерию.\n";

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

cout << "Введите ваш ответ сюда: ";

cin >> choice;

CheckSelect(0, 8, 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 7:

VectorMerge();

break;

case 8:

VectorSort(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();

}

}

// ######################## СЛИЯНИЕ ВЕКТОРОВ ########################

template<typename T>

inline void ConsoleUI<T>::VectorMerge()

{

ChoseVector();

T elem;

vector<T> Merge;

char exit;

cout << "Заполните новый вектор:\n";

for (int i = 1; true; i++)

{

cout << i << ". Введите значение: ";

cin >> elem;

Merge.push\_back(elem);

cout << "Желаете ли вы продолжить? (\"-\" для выхода) : ";

cin >> exit;

if (exit == '-')

break;

}

std::copy(Merge.begin(), Merge.end(), back\_inserter(Vector));

cout << "\nНовый вектор успешно слился со основным.\n";

OperationPass();

}

// ###################### СОРТИРОВКА ВЕКТОРА ########################

template<typename T>

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

{

if (NoVectorData())

return;

SortingMinMax<T> minmax;

SortingMaxMin<T> maxmin;

do

{

ChoseVector();

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

cout << "1. Сортировка по возрастанию.\n";

cout << "2. Сортировка по убыванию.\n";

cout << "0. Отмена.\n";

cout << "Введите ваш ответ сюда: ";

cin >> ch;

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

switch (ch)

{

case 1:

sort(Vector.begin(), Vector.end(), minmax);

cout << endl;

OperationPass();

break;

case 2:

sort(Vector.begin(), Vector.end(), maxmin);

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: ";

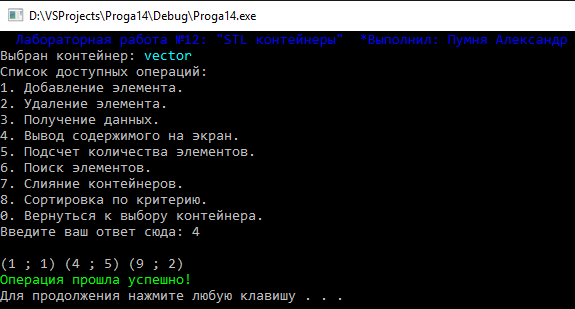
SetColor(11, 0);

cout << "vector\n";

SetColor(7, 0);

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

  
Рисунок 3 – Елементи вектору

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 4 – а) виконання злиття; б) результат злиття | |

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 5 – а) виконання сортування; б) результат сортування | |

ВИСНОВКИ

При виконанні лабораторної роботи набуто практичних навичок щодо використання алгоритмів STL, які змінюють колекцію, та функторів. Розглянуто алгоритми std::sort, std::copy.