Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Тверской государственный университет»

Факультет прикладной математики и кибернетики
Кафедра информатики

Отчет по результатам учебной практики

Направление: 02.03.02 Фундаментальная информатика и
информационные технологии
Специализация: Информатика и компьютерные науки

Выполнил: студентка 26 группы
Николаева Мария Сергеевна

Научный руководитель:
Мальцева Людмила Федоровна

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тверь - 2018

Содержание

1) Реферат (контейнер vector)……………………………………………………3

2) Задача 1…………………………………………………………………………5

3) Задача 2…………………………..…………………………………………....11

1. Реферат (контейнер vector)

vector – шаблонный класс библиотеки STL, реализация динамического массива переменного размера. Элементами вектора могут быть элементы любых типов и сложности, в том числе контейнеры STL (например, vector< stack<int> > или vector< vector< stack<int> > >). Для работы с контейнером vector необходимо подключить соответствующую библиотеку:
#include <vector>.

Некоторые методы шаблонного класса vector:

• **добавление элемента в вектор**: метод **push\_back()**, в который передается добавляемый элемент.

#include <vector>

using namespace std;

int main()

{

    vector<int> numbers;   //пустой вектор из элементов типа int

    numbers.push\_back(5);

    numbers.push\_back(3);

    numbers.push\_back(10);

 return 0;

}

В результате получим вектор numbers = {5, 3, 10}.

• **добавление элементов на определенную позицию**:
 ○ **emplace(pos, value)**: вставляет элемент *value* на позицию, на которую указывает итератор *pos.*

std::vector<int> numbers = { 1, 2, 3, 4, 5 };
auto iter = numbers.begin(); //константный итератор указывает на
 //первый элемент
numbers.emplace(iter + 2, 8); //добавляем 8 после второго элемента
//numbers = {1, 2, 8, 3, 4, 5}

 ○ **insert(pos, n, value)**: вставляет *n* элементов *value,* начиная с позиции, на которую указывает итератор *pos.*

std::vector<int> numbers2 = { 1, 2, 3, 4, 5 };
auto iter2 = numbers2.begin();
numbers2.insert(iter2 + 1, 3, 4); //добавляем после первого элемента
 //три четверки
//numbers2 = {1, 4, 4, 4, 2, 3, 4, 5}

 ○ **insert(pos, begin, end)**: вставляет, начиная с позиции, на которую указывает итератор *pos*, элементы другого контейнера из диапазона между итераторами *begin* и *end*.

std::vector<int> values = { 10, 20, 30, 40, 50 };
std::vector<int> numbers3 = { 1, 2, 3, 4, 5 };
auto iter3 = numbers3.begin();
numbers3.insert(iter3 + 1, values.begin(), values.begin() + 3);
//добавляем в numbers3 после первого элемента элементы из вектора values (с первого по третий)
//numbers3 = {1, 10, 20, 30, 2, 3, 4, 5}

3

• **удаление элементов:**
 ○ **clear()**: отчистить вектор
 ○ **pop\_back()**: удалить последний элемент вектора – метод

std::vector<int> v = { 1,2,3,4 };
v.pop\_back(); //v = {1,2,3}
v.clear(); //v = {}

 ○ **erase(p)**: удаляет элемент, на который указывает итератор p. Возвращает итератор на элемент, следующий после удаленного, или на конец контейнера, если удален последний элемент

std::vector<int> numbers1 = { 1, 2, 3, 4, 5, 6 };
auto iter = numbers1.begin();
numbers1.erase(iter + 2); //удаляем третий элемент
//numbers1 = {1, 2, 4, 5, 6}

• **размер вектора:**
 ○ **size()**: вернет размер вектора
 ○ **empty()**: проверить, путой ли вектор

std::vector<int> numbers = {1, 2, 3};
if(numbers.empty()) cout << "Vector is empty!";
else cout << "Vector has size " << numbers.size();

 ○ **resize(n)**: оставляет в векторе n первых элементов. Если вектор содержит больше элементов, то его размер усекается до n элементов. Если размер вектора меньше n, то добавляются недостающие элементы и инициализируются значением по умолчанию.
 ○ **resize(n, value)**: также оставляет в векторе n первых элементов. Если размер вектора меньше n, то добавляются недостающие элементы со значением value.

std::vector<int> numbers1 = {1, 2, 3, 4, 5, 6};
numbers1.resize(4); //numbers1 = {1, 2, 3, 4}
numbers1.resize(6, 8); //numbers1 = {1, 2, 3, 4, 8, 8}

4

• **сравнение векторов:**

Векторы можно сравнивать. Сравнение контейнеров осуществляется на основании сравнения пар элементов на тех же позициях. Векторы равны, если они содержат одинаковые элементы на тех же позициях. Иначе они не равны:

std::vector<int> v1 = {1, 2, 3};
std::vector<int> v2 = {1, 2, 3};
std::vector<int> v3 = {3, 2, 1};
bool v1v2 = (v1 == v2); // true
bool v1v3 = (v1 != v3); // true
bool v2v3 = (v2 == v3); // false

2) Задача 1

Вариант А, задание №1.
На входе дана произвольная последовательность чисел x1,x2,…
Написать программу, которая
- формирует бинарное дерево поиска;
- подсчитывает число узлов и печатает их снизу вверх
(лев\_п- прав\_п- узел).

#include <string>
#include <stack>
#include <iostream>
#include <fstream>
using namespace std;

struct tree
{
 int inf;
 bool flag = false;
 tree \*left, \*right;
};

class bin\_tree
{
 tree\* root;

public:

 bin\_tree() { root = NULL; }

5

 bool nil\_tree()
 {
 if (root) return false;
 return true;
 }

 void add\_tree(int a)
 {
 tree \*son, \*fath;
 find\_tree(a, son, fath);
 if (son && son->inf == a) return;
 tree\* t = new tree;
 t->inf = a;
 t->left = NULL;
 t->right = NULL;
 if (nil\_tree()) root = t;
 else
 {
 tree \*pt = root, \*pr = pt;
 while (pt)
 {
 pr = pt;
 if (a > pt->inf)
 pt = pt->right;
 else pt = pt->left;
 }
 if (a > pr->inf) pr->right = t;
 else pr->left = t;
 }
 }

 void find\_tree(int a, tree\*&son, tree\*&fath)
 {
 fath = NULL;
 son = root;
 while (son && son->inf != a)
 {
 fath = son;
 if (a > son->inf) son = son->right;
 else son = son->left;
 }
 if (!son) fath = NULL;
 }

6

 void del\_tree\_uz(int a)
 {
 tree \*son, \*fath;
 if (this->nil\_tree())
 {
 cout << "Дерево пусто! Искать тут, к сложалению, нечего.\n";
 return;
 }
 find\_tree(a, son, fath);
 if (!son)
 {
 cout << "Такого элемента нет в нашем дереве поиска.\n";
 return;
 }
 if (son == root)
 {
 if (son->left == NULL && son->right == NULL)
 {
 root = NULL;
 return;
 }
 if (son->left != NULL && son->right == NULL)
 {
 root = root->left;
 return;
 }
 if (son->left == NULL && son->right != NULL)
 {
 root = root->right;
 return;
 }
 }
 if (son->left == NULL && son->right == NULL)
 {
 if (a > fath->inf) fath->right = NULL;
 else fath->left = NULL;
 delete son;
 return;
 }
 if (son->left != NULL && son->right == NULL)
 {
 fath->left = son->left;
 delete son;
 return;
 }
 if (son->left == NULL && son->right != NULL)
 {
 fath->right = son->right;
 delete son;
 return;
 }
 if (son->left != NULL && son->right != NULL)
 {
 tree\* pt = son->right, \*pr = pt;
 while (pt->left)
 {
 pr = pt;
 pt = pt->left;
 }
 son->inf = pt->inf;
 pr->left = NULL;
 if (pt->right && pr != pt) pr->left = pt->right;
 if (pt->right && pr == pt)
 {
 son->right = pt->right;
 return;
 }
 delete pt;
 }
 }

7

 void del\_tree\_p(int a, char side)
 {
 tree \*son, \*fath;
 find\_tree(a, son, fath);
 if (!son) return;
 if (side == 'L') son->left = NULL;
 if (side == 'R') son->right = NULL;
 }

 void view\_tree\_1() //префиксный обход
 {
 stack<tree\*> st;
 tree\* pt = root;
 while (!st.empty() || pt != NULL)
 {
 if (pt != NULL)
 {
 cout << pt->inf << ", ";
 if (pt->right) st.push(pt->right);
 pt = pt->left;
 }
 else
 {
 pt = st.top();
 st.pop();
 }
 }
 }

8

 void view\_tree\_2() //инфиксный обход
 {
 stack<tree\*> st;
 tree\* pt = root;
 while (!st.empty() || pt != NULL)
 {
 if (pt != NULL)
 {
 st.push(pt);
 pt = pt->left;
 }
 else
 {
 pt = st.top();
 st.pop();
 cout << pt->inf << ", ";
 pt = pt->right;
 }
 }
 }

 void view\_tree\_3() //постфиксный обход
 {
 stack<tree\*> st;
 tree\* pt = root;
 st.push(pt);
 while (!st.empty())
 {
 pt = st.top();
 st.pop();
 if (pt)
 {
 if (pt->left == NULL && pt->right == NULL)
 cout << pt->inf << ", ";
 else
 {
 if (pt->flag)
 {
 cout << pt->inf << ", ";
 pt->flag = false;
 }
 else
 {
 pt->flag = true;
 st.push(pt);
 st.push(pt->right);
 st.push(pt->left);
 }
 }
 }
 }
 }
};

9

void individual(string fname)
{
 ifstream file;
 bin\_tree\* treee = new bin\_tree;
 char c;
 int n = 0, k = 0;
 file.open(fname, ios::in);
 if (!file)
 {
 cout << "Sorry, error opening file. Don`t worry and have a nice day! :3\n";
 exit(0);
 }
 file.get(c);
 while (file)
 {
 if (c >= '0' && c <= '9')
 {
 n = n \* 10 + (c - '0');
 file.get(c);
 if (c == ',' && file.peek() == ' ')
 {
 k++;
 treee.add\_tree(n);
 file.get(c); //пропускаем пробел и
 file.get(c); //считываем следующий символ
 }
 }
 else
 {
 cout << "Проверьте входной файл и запустите программу повторно.\n";
 exit(0);
 }
 }
 cout << " ---Количество узлов дерева - " << k << " ---\n";
 treee.view\_tree\_3();
}

**Входной файл**: 15, 11, 28, 26, 4, 6 ,12, 14, 20, 13, 17, 24, 16

10

**Результаты счета**:



3) Задача 2

**Вариант 16**

Написать программу учета книг в библиотеке.

Сведения о книгах содержат: фамилию и инициалы автора, название, год издания, количество экземпляров данной книги в библиотеке.
Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

• добавление данных о книгах, вновь поступающих в библиотеку;
• удаление данных о списываемых книгах;
• выдача сведений о всех книгах, упорядоченных по фамилиям авторов;
• выдача сведений о всех книгах, упорядоченных по годам издания.

Хранение данных организовать с применением контейнерного класса vector.

11

**Исходный код программы:**

#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;

struct librarby
{
 string author, name;
 int year, num;
};

void add\_book(librarby new\_book, vector <librarby>& a)
{
 a.push\_back(new\_book);
}

bool delete\_book(string deleted\_book, vector <librarby>& a)
{
 vector <librarby> a\_new; bool f = false;
 int i, j;
 for (i = 0; i < a.size(); i++)
 {
 if (a[i].name == deleted\_book)
 {
 f = true;
 j = i;
 break;
 }
 }
 if (!f) return false;
 for (i = 0; i < a.size(); i++)
 if (i != j) a\_new.push\_back(a[i]);
 a = a\_new;
 return true;
}

void data\_names(vector <librarby>& a)
{
 setlocale(LC\_ALL, "Russian");
 for (int i = 0; i < a.size() - 1; i++)
 {
 int min = i;
 for (int j = i + 1; j < a.size(); j++)
 {
 if (a[j].author < a[min].author)
 {
 min = j;
 }
 }
 librarby temp = a[i];
 a[i] = a[min];
 a[min] = temp;
 }
 cout << "Информация о книгах, упорядоченная по автору:\n";
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";
 for (int i = 0; i < a.size(); i++)
 cout << i + 1 << ") " << a[i].author << ", " << a[i].name << ", " << a[i].year << " (" << a[i].num << " шт.)" << endl;
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";
}

12

void data\_years(vector <librarby>& a)
{
 setlocale(LC\_ALL, "Russian");
 for (int i = 0; i < a.size() - 1; i++)
 {
 int min = i;
 for (int j = i + 1; j < a.size(); j++)
 {
 if (a[j].year < a[min].year)
 {
 min = j;
 }
 }
 librarby temp = a[i];
 a[i] = a[min];
 a[min] = temp;
 }
 cout << "Информация о книгах, упорядоченная по году издания:\n";
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";
 for (int i = 0; i < a.size(); i++)
 cout << i + 1 << ") " << a[i].author << ", " << a[i].name << ", " << a[i].year << " (" << a[i].num << " шт.)" << endl;
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";
}

13

void fill\_base(string fname, vector <librarby>& a)
{
 librarby added\_book;
 string title, surname, inic; char b;
 int \_year, \_num;
 ifstream file("text.txt");
 if (!file)
 {
 cout << "Ошибка при открытии файла." << endl;
 exit(1);
 }
 while (file)
 {
 title = "";
 file.get(b);
 title += b;
 file.get(b);
 while (b != '"')
 {
 title += b;
 file.get(b);
 }
 file >> surname;
 file >> inic;
 added\_book.name = title + "\"";
 added\_book.author = surname + " " + inic;
 file >> \_year;
 file >> \_num;
 added\_book.year = \_year;
 added\_book.num = \_num;
 add\_book(added\_book, a);
 file.get(b);
 }
}

int main()
{
 setlocale(LC\_ALL, "Russian");
 vector <librarby> a;
 int n; string title; bool fl;
 char b;
 cout << "\n Добрый день! Прекрасно выглядите! Вы находитесь в системе учета книг." << endl;
 while (1)
 {
 title = "";
 cout << "\n\n Выберите дальнейшее действие:" << endl;
 cout << " 1) Добавить данные о книгах из файла" << endl;
 cout << " 2) Удалить данные о книге" << endl;
 cout << " 3) Вывести информацию обо всех книгах (упор. по автору)" << endl;
 cout << " 4) Вывести информацию обо всех книгах (упор. по году издания)" << endl;
 cout << " 5) Выход" << endl;
 cout << "\n Введите номер выбранного действия: ";
 cin >> n;
 cout << "\n\n";
 switch (n)
 {
 case 1:
 fill\_base("text.txt", a);
 cout << " ---Книги успешно добавлены в базу---\n";
 break;
 case 2:
 cout << " Введите название книги, данные о которой хотите удалить: ";
 cin >> b;
 getline(cin, title);
 title = "\"" + title;
 fl = delete\_book(title, a);
 if (fl) cout << "\n\n ---Данные о книге " << title << " успешно удалены---\n\n";
 else cout << "\n\n --ВНИМАНИЕ: Проверьте правильность введенных данных - книга с таким названием отсутствует в базе--\n";
 break;
 case 3:
 data\_names(a);
 break;
 case 4:
 data\_years(a);
 break;
 case 5:
 cout << " ---Благодарим за использование нашей системы. До свидания!---\n\n" << endl;
 system("pause");
 exit(1);
 break;
 default:
 cout << " --Пожалуйста, проверьте и повторите ввод: этот пункт отсутствует--\n";
 break;
 }
 }
 system("pause");
 return 0;
}

14

15

**Входной файл:**

"Title 1" Nikolaeva M.S. 1998 23
"Title 2" Naikolaeva M.S. 1999 234
"Title 5" Nbikolaeva M.S. 1979 234
"Title 4" Nikolaeva M.S. 1499 234

**Результаты счета:**

****

16







17