Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

кафедра Апу

**ОТЧЕТ**

**по лабораторной работе №3**

**«Последовательный и бинарный поиск**

**в одномерном массиве»**

Выполнил: Панфёров М. А.

Группа №3371

Преподаватель: Ряскова Е. Б.

Санкт-Петербург

2024

Постановка задачи

Требуется в программу из лабораторной работы №2 добавить возможность осуществления поиска в одномерном отсортированном массиве структур. Поиск идёт по тому же значению, на основе которого осуществляется сортировка, в данном случае – по коду. Релизовать нужно двумя способами:

* Прямой (последовательный) поиск
* Бинарный поиск

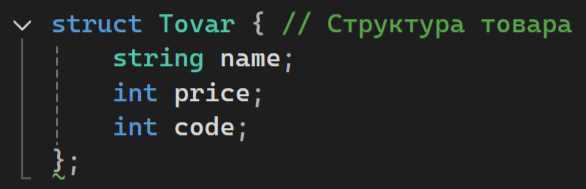
Листинг программы

Рис. 1 – структура данных товара.

На рисунке 1 показана структура данных товара. Он в себе содержит данные от его названии, сцены и кода.

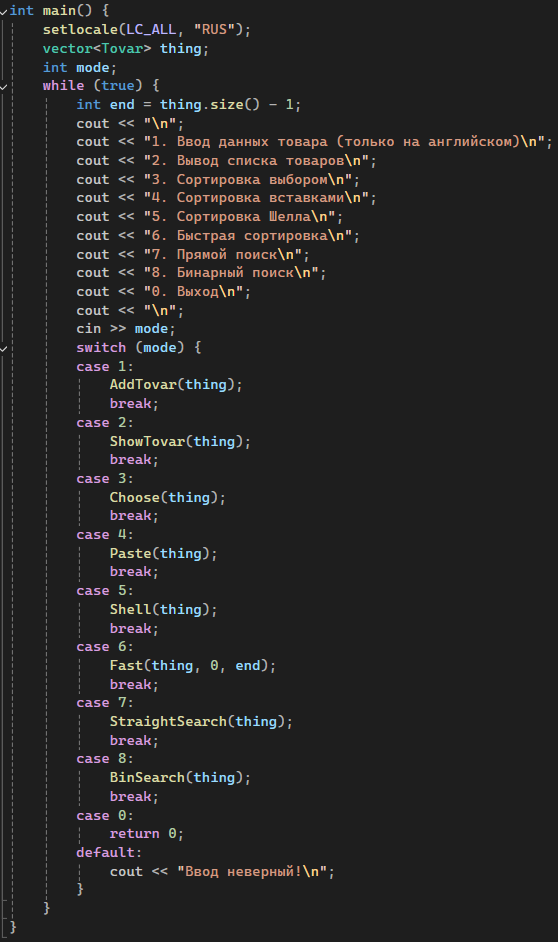


Рис. 2 – главная функция.

На рисунке 2 показана глафная функция. В ней создаётся массив структур и находится меню. После идёт выбор пункта меню, если указать число, которое не указано в пункте меню, то программа сообщит, что ввод неверен и будет ожидать ввода верного числа. Для работы быстрой сортировки была создана функция для вызова последнего элемента.

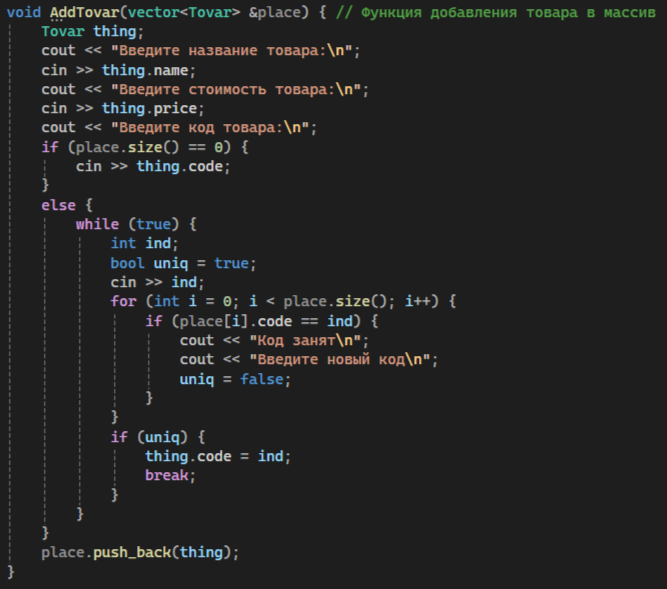


Рис. 3 – функция добавления товара в массив

На рисунке 3 показана функция добавления товара в массив. На вход подаётся массив. После создаётся “thing” – товар, которому после присваивается название и стоимость. Код присваивается иным способом. Когда массив пустой, код просто присваивается, но когда в массиве есть элементы, то идёт проверка уникальности кода. Если код повторяется – то пограмма сообщит, что код занят и нужно ввести новый, иначе код присвоится и товар будет добавлен в массив.

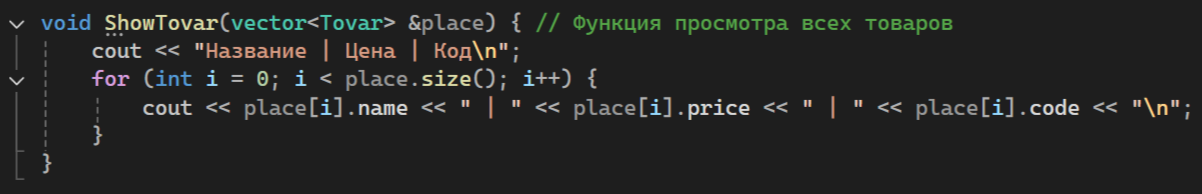


Рис. 4 – функция вывода содежимого массива

На рисунке 4 показана функция вывода данных массива. На вход подаётся массив. Вывод происходит в виде некоторой таблицы: «Название | Цена | Код». Вывод реализован при помощи цикла, в котором каждый i-тый элемент выводится.

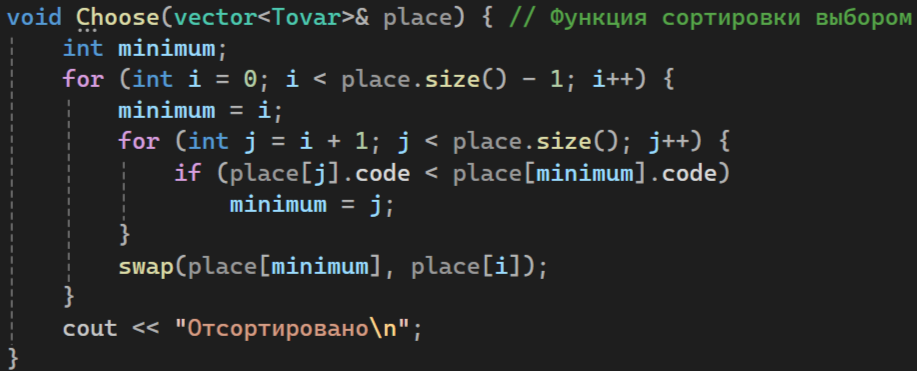


Рис. 5 – функция сортировки выбором

На рисунке показана функция, где реализована сортировка выбором. На вход подаётся массив. Создаётся переменная «minimum» - она определяет положение элемента, нужна для сравнения последующих элементов с текущим. Далее идёт цикл, который сравивает текущее значение, с «minimum», если текущее значение меньше «minimum», то текущее значение становится им и они меняются местами. И так до того, как массив не будет отсортирован.

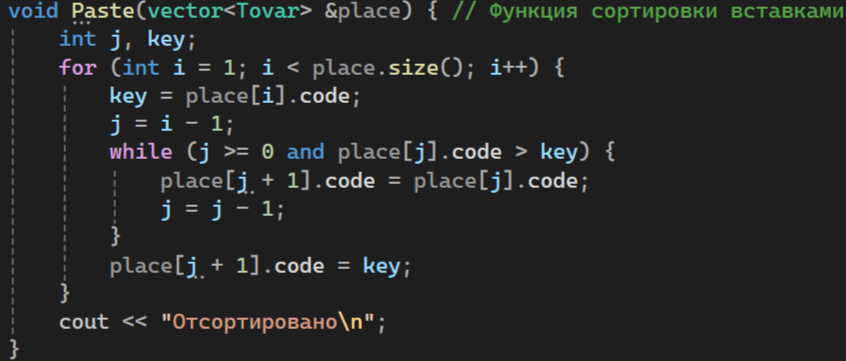
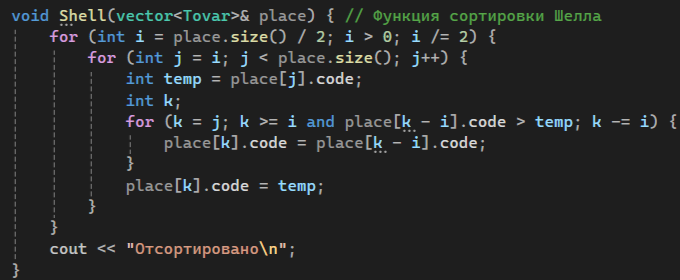


Рис. 6 – функция сортировки вставками

На рисунке 6 показана функция сортировка вствками. На вход подаётся массив. Создаются переменные «j» и «key». В начале цикла «key» принимает значение текущего элемента, а «j» - предыдущего. После идёт цикл, где следующий элемент заменяется текущим, после «key» становится следующим элементом. И так до того как массив не будет отсортирован.

Рис. 7 – функция сортировки Шелла

На рисунке 7 показана сортировка Шелла. Сама сортировка предствавляет собой улучшенную сортировку вставками, только элементы сравниваются с определённым интервалом.

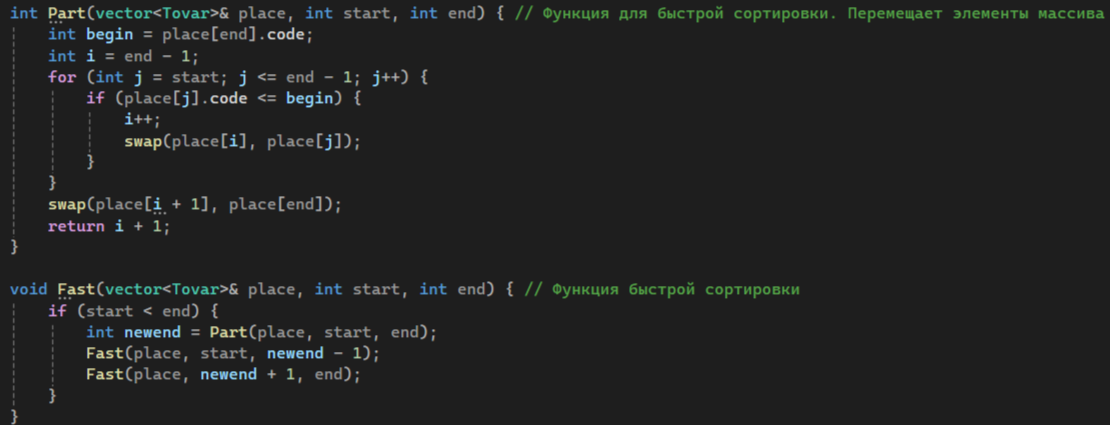


Рис 8. – реализация быстрой сортировки

На рисунке 8 показана реализация быстрой сортировки. Для неё было написано 2 функции – Part и Fast. Fast – это функция, при помощи которой эта функция вызывается. В ней реализованы рекурсивные функции, в которых элементы перемещаются в зависимости от размера выбранного числа. В Part происходит перемещение элементов.

Рис. 9 – Функция прямого поиска

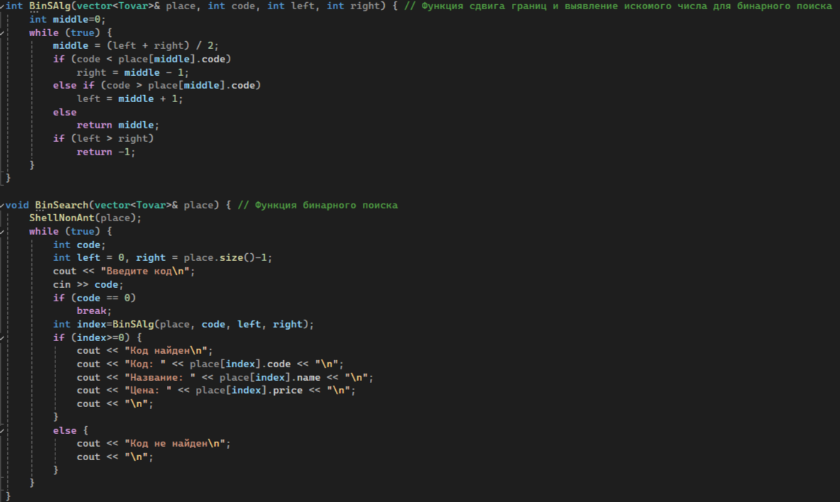
На рисунке 9 показана функция прямого поиска. На вход подаётся массив структур. В цикле реализован сам поиск. Сам прямой поиск представляет собой сравнение ищущего с элементами массива. В циклу в начале нужно ввести код, после идёт цикл, где ведённый код сравнивается с кодами структур массива. Если было обнаружено совпадение, то будет показан найденный товар, иначе сообщит, что код не найден. В обоих случаях программа предложит ввести ещё коды, а при вводе нуля – выйти.

Рис. 10 – Функции бинарного поиска

На рисунке 10 показаны функции бинарного поиска. Поиск состоит из двух функций – функция, отвечающая за входные данные, ввод кода и вывод результата, и функция самого бинарного поиска. Бинарная сортировка делит массив попалам и проверяет границы и эти границы сдвигает, что в итоге остаётся нужный элемент. В хорошем случае функция вернёт индекс элемента и во второй функции будет показан товар, иначе будет чсообщено, что код не найден, в обоихслучаях программа предложит ввести ещё коды, а при вводе нуля – выйти. Также важно отметить, что этот поиск работает только в отсортированном массиве, поэтому перед сортировкой массив автоматический сортируется.

Код и пример работы программы

AiSD\_Laba3.cpp:

#include "iostream"

#include "string"

#include "vector"

using namespace std;

// Дополнение кода программы двумя вариантами поиска: прямая и бинарная

struct Tovar { // Структура товара

string name;

int price;

int code;

};

void AddTovar(vector<Tovar>& place) { // Функция добавления товара в массив

Tovar thing;

cout << "Введите название товара:\n";

cin >> thing.name;

cout << "Введите стоимость товара:\n";

cin >> thing.price;

cout << "Введите код товара:\n";

if (place.size() == 0) {

cin >> thing.code;

}

else {

while (true) {

int ind;

bool uniq = true;

cin >> ind;

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

if (place[i].code == ind) {

cout << "Код занят\n";

cout << "Введите новый код\n";

uniq = false;

}

}

if (uniq) {

thing.code = ind;

break;

}

}

}

place.push\_back(thing);

}

void ShowTovar(vector<Tovar>& place) { // Функция просмотра всех товаров

cout << "Название | Цена | Код\n";

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

cout << place[i].name << " | " << place[i].price << " | " << place[i].code << "\n";

}

}

void Choose(vector<Tovar>& place) { // Функция сортировки выбором

int minimum;

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

minimum = i;

for (int j = i + 1; j < place.size(); j++) {

if (place[j].code < place[minimum].code)

minimum = j;

}

swap(place[minimum], place[i]);

}

cout << "Отсортировано\n";

}

void Paste(vector<Tovar>& place) { // Функция сортировки вставками

int j, key;

for (int i = 1; i < place.size(); i++) {

key = place[i].code;

j = i - 1;

while (j >= 0 and place[j].code > key) {

place[j + 1].code = place[j].code;

j = j - 1;

}

place[j + 1].code = key;

}

cout << "Отсортировано\n";

}

void Shell(vector<Tovar>& place) { // Функция сортировки Шелла

for (int i = place.size() / 2; i > 0; i /= 2) {

for (int j = i; j < place.size(); j++) {

int temp = place[j].code;

int k;

for (k = j; k >= i and place[k - i].code > temp; k -= i) {

place[k].code = place[k - i].code;

}

place[k].code = temp;

}

}

cout << "Отсортировано\n";

}

int Part(vector<Tovar>& place, int start, int end) { // Функция для быстрой сортировки. Перемещает элементы массива

int begin = place[end].code;

int i = end - 1;

for (int j = start; j <= end - 1; j++) {

if (place[j].code <= begin) {

i++;

swap(place[i], place[j]);

}

}

swap(place[i + 1], place[end]);

return i + 1;

}

void Fast(vector<Tovar>& place, int start, int end) { // Функция быстрой сортировки

if (start < end) {

int newend = Part(place, start, end);

Fast(place, start, newend - 1);

Fast(place, newend + 1, end);

}

}

void ShellNonAnt(vector<Tovar>& place) { // Функция сортировки Шелла

for (int i = place.size() / 2; i > 0; i /= 2) {

for (int j = i; j < place.size(); j++) {

int temp = place[j].code;

int k;

for (k = j; k >= i and place[k - i].code > temp; k -= i) {

place[k].code = place[k - i].code;

}

place[k].code = temp;

}

}

}

void StraightSearch(vector<Tovar>& place) { // Функция прямого поиска

cout << "Для выхода введите 0\n";

while (true) {

int code;

bool finded = false;

cout << "Введите код\n";

cin >> code;

if (code == 0)

break;

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

if (place[i].code==code) {

cout << "Код найден\n";

cout << "Код: " << place[i].code << "\n";

cout << "Название: " << place[i].name << "\n";

cout << "Цена: " << place[i].price << "\n";

cout << "\n";

finded = true;

break;

}

}

if (!finded) {

cout << "Код не найден\n";

cout << "\n";

}

}

}

int BinSAlg(vector<Tovar>& place, int code, int left, int right) { // Функция сдвига границ и выявление искомого числа для бинарного поиска

int middle=0;

while (true) {

middle = (left + right) / 2;

if (code < place[middle].code)

right = middle - 1;

else if (code > place[middle].code)

left = middle + 1;

else

return middle;

if (left > right)

return -1;

}

}

void BinSearch(vector<Tovar>& place) { // Функция бинарного поиска

ShellNonAnt(place);

while (true) {

int code;

int left = 0, right = place.size()-1;

cout << "Введите код\n";

cin >> code;

if (code == 0)

break;

int index=BinSAlg(place, code, left, right);

if (index>=0) {

cout << "Код найден\n";

cout << "Код: " << place[index].code << "\n";

cout << "Название: " << place[index].name << "\n";

cout << "Цена: " << place[index].price << "\n";

cout << "\n";

}

else {

cout << "Код не найден\n";

cout << "\n";

}

}

}

int main() {

setlocale(LC\_ALL, "RUS");

vector<Tovar> thing;

int mode;

while (true) {

int end = thing.size() - 1;

cout << "\n";

cout << "1. Ввод данных товара (только на английском)\n";

cout << "2. Вывод списка товаров\n";

cout << "3. Сортировка выбором\n";

cout << "4. Сортировка вставками\n";

cout << "5. Сортировка Шелла\n";

cout << "6. Быстрая сортировка\n";

cout << "7. Прямой поиск\n";

cout << "8. Бинарный поиск\n";

cout << "0. Выход\n";

cout << "\n";

cin >> mode;

switch (mode) {

case 1:

AddTovar(thing);

break;

case 2:

ShowTovar(thing);

break;

case 3:

Choose(thing);

break;

case 4:

Paste(thing);

break;

case 5:

Shell(thing);

break;

case 6:

Fast(thing, 0, end);

break;

case 7:

StraightSearch(thing);

break;

case 8:

BinSearch(thing);

break;

case 0:

return 0;

default:

cout << "Ввод неверный!\n";

}

}

}

Пример работы программы:

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

1

Введите название товара:

Sausage

Введите стоимость товара:

301

Введите код товара:

123

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

1

Введите название товара:

Meat

Введите стоимость товара:

1284

Введите код товара:

2345

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

1

Введите название товара:

Cheese

Введите стоимость товара:

186

Введите код товара:

54

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

1

Введите название товара:

Bitter

Введите стоимость товара:

176

Введите код товара:

45

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

1

Введите название товара:

kefir

Введите стоимость товара:

103

Введите код товара:

357

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

1

Введите название товара:

Yogurt

Введите стоимость товара:

38

Введите код товара:

2890

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

2

Название | Цена | Код

Sausage | 301 | 123

Meat | 1284 | 2345

Cheese | 186 | 54

Bitter | 176 | 45

kefir | 103 | 357

Yogurt | 38 | 2890

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

5

Отсортировано

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

2

Название | Цена | Код

Sausage | 301 | 45

Meat | 1284 | 54

Cheese | 186 | 123

Bitter | 176 | 357

kefir | 103 | 2345

Yogurt | 38 | 2890

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

8

Введите код

122

Код не найден

Введите код

357

Код найден

Код: 357

Название: Bitter

Цена: 176

Введите код

2890

Код найден

Код: 2890

Название: Yogurt

Цена: 38

Введите код

2567

Код не найден

Введите код

2345

Код найден

Код: 2345

Название: kefir

Цена: 103

Введите код

0

1. Ввод данных товара (только на английском)

2. Вывод списка товаров

3. Сортировка выбором

4. Сортировка вставками

5. Сортировка Шелла

6. Быстрая сортировка

7. Прямой поиск

8. Бинарный поиск

0. Выход

0

Вывод: я узнал принцип работы бинарного и прямого поиска, тепер я могу использовать данные методы поиска в своих проектах.