Министерство образования Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №9**

**По дисциплине «Основы алгоритмизации и программирования»**

**Тема: «Сортировка и поиск»**

Выполнил: Шиковец Е.А.

Группа 21-ИТ-1

Проверила: Пантелейко А.Ф

Преподаватель-стажер кафедры ТП

Полоцк 2022

**Цель работы:** изучить методы организации списочных структур в динамической памяти. Реализовать алгоритмы помещения и изъятия элементов из стека, дека или очереди.

**Вариант № 4**

1. Организуйте выгрузку данных из Вашей структуры данных в динамический массив.

2. Организуйте выгрузку данных из динамического массива в Вашу структуру данных.

3. Организуйте сортировку элементов массива по любому полю любым алгоритмом.

4. Организуйте медленный поиск записей в Вашей структуре данных (или быстрый поиск в предварительно выгруженном и отсортированном динамическом массиве) по любому полю.

5. Добавьте написанные Вами функции в курсовую работу.

#include <iostream>

#include <stdlib.h>

#include <iomanip>

#include <string.h>

using namespace std;

struct T\_inf // tenant info

{

// данные квартир

char \*street;

int house\_num;

int apart\_num;

int area;

int rooms;

// данные жильцов

char \*name;

char \*birth\_date;

char \*reg\_date; // дата прописки

char \*disch\_date; // дата выписки

char \*rel; // отношение к владельцу квартиры

};

struct Array

{

// данные квартир

char \*street;

int house\_num;

int apart\_num;

int area;

int rooms;

// данные жильцов

char \*name;

char \*birth\_date;

char \*reg\_date; // дата прописки

char \*disch\_date; // дата выписки

char \*rel; // отношение к владельцу квартиры

};

T\_inf \*add\_tenant(struct T\_inf \*tenants, int pos)

{

system("CLS");

cout << "Size: " << pos << endl;

cout << "-- Adding a tenant --" << endl;

// информация о квартире

cout << "Street: ";

tenants[pos].street = new char[20];

cin >> tenants[pos].street;

cin.ignore();

cout << "House number: ";

cin >> tenants[pos].house\_num;

cout << "Apartment number: ";

cin >> tenants[pos].apart\_num;

cout << "Apartment area: ";

cin >> tenants[pos].area;

cout << "Rooms: ";

cin >> tenants[pos].rooms;

// информация о жителе

cout << "Name: ";

tenants[pos].name = new char[20];

cin >> tenants[pos].name;

cout << "Date of birth: ";

tenants[pos].birth\_date = new char[20];

cin >> tenants[pos].birth\_date;

cout << "Regestration date: ";

tenants[pos].reg\_date = new char[20];

cin >> tenants[pos].reg\_date;

cout << "Discharge date: "; // дата выписки

tenants[pos].disch\_date = new char[20];

cin >> tenants[pos].disch\_date;

cout << "Relations: "; // отношение к владельцу квартиры

tenants[pos].rel = new char[20];

cin >> tenants[pos].rel;

system("CLS");

cout << tenants[pos].name << " was succesfuly added!\n\n";

return tenants; // возвращаем новый массив

}

void get\_data(struct T\_inf \*tenants, int pos)

{

if (!pos)

{

cout << "Structure is empty\n";

return;

}

int count = 0; // количество проживающих в однокомнатных квартирах

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

{

if (tenants[i].rooms == 1)

{

count++;

}

}

cout << "Number of tenants in 1 room apartments: " << count << endl;

}

void output(struct T\_inf \*tenants, int pos) // вывод в формате списка

{

if (!pos)

{

cout << "Structure is empty\n";

return;

}

cout << "Structure output:\n";

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

{

cout << i << ". ";

cout << "Street: " << tenants[i].street << " House num: " << tenants[i].house\_num << " Apart num: " << tenants[i].apart\_num << " Area: " << tenants[i].area;

cout << " Rooms: " << tenants[i].rooms << " Name: " << tenants[i].name << " Birth date: " << tenants[i].birth\_date << " Regestration: " << tenants[i].reg\_date;

cout << " Disch date: " << tenants[i].disch\_date << " Relations: " << tenants[i].rel;

cout << endl;

}

}

void table(struct T\_inf \*tenants, int pos) // вывод в табличном формате

{

if (!pos)

{

cout << "Structure is empty\n";

return;

}

cout << "Structure table:\n";

cout << setw(15) << "Street" << setw(15) << "House num" << setw(15) << "Apart num" << setw(15) << "Area" << setw(15) << "Rooms" << setw(15) << "Name" << setw(15) << "Birth date" << setw(15) << "Reg" << setw(15) << "Disch" << setw(15) << "Rel" << endl;

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

{

cout << setw(15) << tenants[i].street << setw(15) << tenants[i].house\_num << setw(15) << tenants[i].apart\_num << setw(15) << tenants[i].area << setw(15) << tenants[i].rooms << setw(15) << tenants[i].name << setw(15) << tenants[i].birth\_date << setw(15) << tenants[i].reg\_date << setw(15) << tenants[i].disch\_date << setw(15) << tenants[i].rel;

cout << endl;

}

}

Array \*sort(struct Array \*array, int arr\_pos) // сортировка по house\_num

{

for (int i = 0; i < arr\_pos - 1; i++)

{

for (int j = 0; j < arr\_pos - 1; j++)

{

if (array[j].house\_num > array[j + 1].house\_num)

{

Array temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

return array;

}

void find(struct Array \*array, int arr\_pos, int house\_num) // поиск

{

bool is\_found = 0;

for (int i = 0; i < arr\_pos; i++)

{

if (array[i].house\_num == house\_num)

{

cout << setw(15) << "Street" << setw(15) << "House num" << setw(15) << "Apart num" << setw(15) << "Area" << setw(15) << "Rooms" << setw(15) << "Name" << setw(15) << "Birth date" << setw(15) << "Reg" << setw(15) << "Disch" << setw(15) << "Rel" << endl;

cout << setw(15) << array[i].street << setw(15) << array[i].house\_num << setw(15) << array[i].apart\_num << setw(15) << array[i].area << setw(15) << array[i].rooms << setw(15) << array[i].name << setw(15) << array[i].birth\_date << setw(15) << array[i].reg\_date << setw(15) << array[i].disch\_date << setw(15) << array[i].rel;

cout << endl;

is\_found = 1;

}

}

if (!is\_found)

{

cout << "We`ve not found " << house\_num << endl;

}

return;

}

int main()

{

system("CLS");

system("Color 09");

struct T\_inf \*tenants = new T\_inf[10];

struct Array \*array = new Array[10];

int n = -1, pos = 0, arr\_pos = 0;

while (n != 0)

{

cout << "-- Main menu --\n";

cout << "1. Add a tenant\n";

cout << "2. Get data\n";

cout << "3. Output\n";

cout << "4. Table\n";

cout << "5. Clear\n";

cout << "6. Load to array\n";

cout << "7. Print array\n";

cout << "8. Load from array\n";

cout << "9. Sort\n";

cout << "10. Search\n";

cout << "0. To exit\n";

cin >> n;

switch (n)

{

case 1:

system("CLS");

tenants = add\_tenant(tenants, pos);

pos++;

break;

case 2:

system("CLS");

get\_data(tenants, pos);

break;

case 3:

system("CLS");

output(tenants, pos);

break;

case 4:

system("CLS");

table(tenants, pos);

break;

case 5:

{

system("CLS");

delete[] tenants;

struct T\_inf \*tenants = new T\_inf[10];

pos = 0;

break;

}

case 6:

system("CLS");

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

{

array[i].street = tenants[i].street;

array[i].house\_num = tenants[i].house\_num;

array[i].apart\_num = tenants[i].apart\_num;

array[i].area = tenants[i].area;

array[i].rooms = tenants[i].rooms;

array[i].name = tenants[i].name;

array[i].birth\_date = tenants[i].birth\_date;

array[i].reg\_date = tenants[i].reg\_date;

array[i].disch\_date = tenants[i].disch\_date;

array[i].rel = tenants[i].rel;

}

arr\_pos = pos;

break;

case 7:

system("CLS");

if (!arr\_pos)

{

cout << "Structure is empty\n";

}

cout << "Array table:\n";

cout << setw(15) << "Street" << setw(15) << "House num" << setw(15) << "Apart num" << setw(15) << "Area" << setw(15) << "Rooms" << setw(15) << "Name" << setw(15) << "Birth date" << setw(15) << "Reg" << setw(15) << "Disch" << setw(15) << "Rel" << endl;

for (int i = 0; i < arr\_pos; i++)

{

cout << setw(15) << array[i].street << setw(15) << array[i].house\_num << setw(15) << array[i].apart\_num << setw(15) << array[i].area << setw(15) << array[i].rooms << setw(15) << array[i].name << setw(15) << array[i].birth\_date << setw(15) << array[i].reg\_date << setw(15) << array[i].disch\_date << setw(15) << array[i].rel;

cout << endl;

}

break;

case 8:

system("CLS");

for (int i = 0; i < arr\_pos; i++)

{

tenants[i].street = array[i].street;

tenants[i].house\_num = array[i].house\_num;

tenants[i].apart\_num = array[i].apart\_num;

tenants[i].area = array[i].area;

tenants[i].rooms = array[i].rooms;

tenants[i].name = array[i].name;

tenants[i].birth\_date = array[i].birth\_date;

tenants[i].reg\_date = array[i].reg\_date;

tenants[i].disch\_date = array[i].disch\_date;

tenants[i].rel = array[i].rel;

}

pos = arr\_pos;

break;

case 9:

system("CLS");

sort(array, arr\_pos);

break;

case 10:

system("CLS");

sort(array, arr\_pos);

int house\_num;

cout << "Input house number: ";

cin >> house\_num;

system("CLS");

find(array, arr\_pos, house\_num);

break;

case 0:

break;

default:

system("CLS");

cout << "Check your input\n";

break;

}

}

system("Color 07");

system("CLS");

delete[] tenants;

delete[] array;

return 0;

}

**Вывод:**

Изучены принципы работы сортировки и поиска. Разработана программа с использованием сортировки и поиска.