**Министерство образования и науки Российской Федерации**

**Пермское федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский Национальный Исследовательский Политехнический Университет»**

**Электротехнический факультет**

**Кафедра «Информационные технологии и автоматизированные системы»**

**ОТЧЁТ**

По лабораторной работе №12 на тему

«Алгоритмы поиска»

Вариант №11

Выполнил студент группы РИС-20-1б

Шумилов Лев Сергеевич

Проверил доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Цель работы** –научиться работать с составными динамическими типами данных и получить практические навыки работы с такими типами данных, как:

1. Однонаправленные списки;
2. Двунаправленные списки;
3. Стеки;
4. Очереди;

**Постановка задачи**

Задача – реализовать на языке C++ программу, которая реализует следующий функционал:

* Работа со структурами, заданными в варианте;
* Линейный поиск в структурах;
* Интерполяционный поиск в структурах;
* Линейный поиск подстроки в строке;
* Консольное меню для взаимодействия с программой;

Исходные данные для **варианта №11**:

Структура «Человек», содержащая следующие поля:

1. Фамилия, имя, отчество – полное имя;
2. Номер паспорта – число;
3. Номер телефона – строка;

Линейный и интерполяционный поиск осуществлять через вводимый пользователем ключ **по номеру паспорта**.

**Анализ задачи**

Для решения задачи были использованы следующие средства:

1. Язык программирования C++ (Microsoft Visual C++)
2. Текстовый редактор Microsoft Visual Studio Code
3. Система контроля версий Git

Вышеописанная структура в языке программирования C++ объявляется следующим образом:

struct Person

{

string fullName;

int passportNumber;

string phoneNumber;

};

Поскольку программа будет работать только с одним массивом структур и одной строкой, то (для удобства) они будут объявлены как глобальные переменные:

1. Количество элементов массива;
2. Массив с данными о людях;
3. Строка;

int \_personsCount;

Person\* \_persons;

string \_string;

Длина массива и номер паспорта являются натуральными числами. Соответственно, при вводе целых чисел необходимо производить проверку на натуральность (если число больше нуля, то оно натуральное). Для ввода натуральных чисел была реализована функция ReadNaturalNum, которая решает данную задачу (если введённое число не натуральное, функция попросит новый ввод).

int ReadNaturalNum()

{

int x;

do cin >> x; while (x <= 0);

return x;

}

Для того, чтобы организовать ввод данных о новых людях, была реализована функция InputNewPersons. Функция просит ввести количество элементов нового массива и заполнить данных во всех элементах.

Чтобы вводить строки с пробелами, используется функция getline, которая принимает строку на ввод пока не будет нажат enter.

Алгоритм функции работает по следующим шагам:

1. Вывод сообщения о просьбе ввести количество элементов нового массива;
2. Принятие на ввод числа с клавиатуры (натуральное число);
3. Вывод сообщения в вводе конкретного поля каждого элемента;
4. Принятие на ввод элементов полей;
5. Запись данных в массив \_persons;

void InputNewPersons()

{

cout << "Enter the count of persons: ";

\_personsCount = ReadNaturalNum();

\_persons = new Person[\_personsCount];

for (int i = 0; i < \_personsCount; i++)

{

cout << endl << "Full name: ";

cin.ignore();

getline(cin, \_persons[i].fullName);

cout << "Passport number: ";

cin >> \_persons[i].passportNumber;

cout << "Phone number: ";

cin.ignore();

getline(cin, \_persons[i].phoneNumber);

}

cout << endl;

}

Ввод данных в строку осуществляется при помощи функции InputNewString.

void InputNewString()

{

getline(cin, \_string);

}

Вывод массива организовывается при помощи функций PrintAllPersons, которая вызывает функцию PrintPersons, принимающую в качестве аргумента массив, который необходимо вывести. PrintPersons в цикле for на каждой итерации вызывает функцию PrintPerson, которая выводит на экран все данные одного элемента.

void PrintPerson(Person person, int index)

{

cout << endl << "Person #" << index + 1 << ":\n";

cout << " - Full name: " << person.fullName << endl;

cout << " - Passport number: " << person.passportNumber << endl;

cout << " - Phone number: " << person.phoneNumber << endl << endl;

}

void PrintPersons(Person\* persons)

{

for (int i = 0; i < \_personsCount; i++)

PrintPerson(persons[i], i);

}

void PrintAllPersons()

{

PrintPersons(\_persons);

}

**Реализация алгоритмов поиска:**

1. Линейный поиск

Линейный поиск реализуется при помощи функции DoLineSearch, которая принимает на ввод ключ поиска и вызывает основную функцию поиска по данному ключу LinearFindPeople.

Данная функция проходит по каждому элементу через цикл for; если поле элемента совпадает с ключом, значит элемент найден и его необходимо вывести на экран. Выводятся все найденные элементы. Также присутствует флаг itemFound, сигнализирующий о том, был ли найден хоть один элемент по данному ключу или нет; если ни один элемент не совпал с ключом, то выводится сообщение о том, что массив не содержит элементов с заданным ключом.

void LinearFindPeople(Person\* persons, int searchKey)

{

bool itemFound = false;

for (int i = 0; i < \_personsCount; i++)

if (persons[i].passportNumber == searchKey)

{

itemFound = true;

cout << "Item \"" << searchKey << "\" found: "

<< "index = " << i + 1 << ";\n";

PrintPerson(persons[i], i);

}

if (!itemFound)

cout << "The list has no items with the key;" << endl;

}

void DoLineSearch()

{

cout << "Enter the key of searching (passport number): ";

int key = ReadNaturalNum();

LinearFindPeople(\_persons, key);

}

1. **Интерполяционный поиск:**

Данный метод поиска является одним для самых быстрых для нахождения чисел в массиве. Высокая скорость поиска достигается благодаря использованию математических методов поиска. Их результатом является формула ниже, где *middle* – индекс среднего элемента, *left* – индекс левого элемента, *right* – индекс правого элемента, а *array* – массив, в котором производится поиск.

Данный метод применим только к сортированным массивом. Соответственно перед тем, как производить поиск данным методом, массив следует отсортировать. Для примера будет использоваться метод сортировки выбором, но с тем же успехом можно использовать любой другой метод.

void SelectionSort(Person\*& persons)

{

int j = 0;

Person temp;

for (int i = 0; i < \_personsCount; i++)

{

j = i;

for (int k = i; k < \_personsCount; k++)

if (persons[j].passportNumber > persons[k].passportNumber)

j = k;

temp = persons[i];

persons[i] = persons[j];

persons[j] = temp;

}

}

void InterpolFindPeople(Person\* persons, int searchKey)

{

bool itemFound = false;

int

middle,

left = 0,

right = \_personsCount,

startIndex = -1;

while ((persons[left].passportNumber <= searchKey) && (persons[right].passportNumber >= searchKey))

{

middle = left + (searchKey - persons[left].passportNumber) \* (right - left) / (persons[right].passportNumber - persons[left].passportNumber);

if (persons[middle].passportNumber < searchKey)

left = middle + 1;

else if (persons[middle].passportNumber > searchKey)

right = middle - 1;

else

{

startIndex = middle;

itemFound = true;

break;

}

}

if (!itemFound)

cout << "The list has no items with the key;" << endl;

else

{

//поиск элементов (если найденных элементов больше одного

for (int i = startIndex; i < \_personsCount; i++)

{

if (!PrintCompare(persons, i, searchKey))

break;

}

for (int i = startIndex - 1; i >= 0; i--)

{

if (!PrintCompare(persons, i, searchKey))

break;

}

}

}

Функция DoInterpolationSearch, отвечает за использования данного метода: принимает на ввод ключ для поиска, сортирует массив и выводит найденные элементы на экран.

void DoInterpolationSearch()

{

cout << "Enter the key of searching (passport number): ";

int key = ReadNaturalNum();

SelectionSort(\_persons);

InterpolFindPeople(\_persons, key);

}

1. **Линейный поиск подстроки в строке:**

Линейный поиск подстроки в строке работает по схожему принципу с обычным линейным поиском.

* Идёт проход от начала до конца строки;
* Идёт сравнения одного символа строки и первого символа подстроки;
* Если символы совпали, то начинается накопления подстроки;
* Если последующие символы совпадают, то строка продолжает накапливаться, иначе подстрока обнуляется;

Если строка не найдена, то выводится соответствующее сообщение.

Функция FindSubstring принимает на ввод подстроку, которую необходимо найти и вызывает основную функцию нахождения подстроки FindSubstring, которая принимает два аргумента: строку, в который происходит поиск, подстроку, которую необходимо найти.

void FindSubstring(string& baseString, string& substring)

{

string foundSubstring = "";

int index = 0;

for (int i = 0; i < baseString.length(); i++)

{

if (baseString[i] == substring[index])

{

foundSubstring += baseString[i];

if (index == substring.length() - 1)

break;

index++;

}

else

{

foundSubstring = "";

index = 0;

}

}

if (foundSubstring == substring)

cout << "Substring \"" << substring << "\" has been found!" << endl;

else

cout << "The string doesn't contain \""

<< substring << "\"!\n";

}

void FindSubstring()

{

cout << "Enter the substring, which you want to find: ";

string subString;

getline(cin, subString);

FindSubstring(\_string, subString);

}

**Реализация меню:**

Меню в программе будет работать по следующей модели:

* Принимается на ввод строка – имя команды, которую необходимо вызвать;
* Строка проверяется на равенство с соответствующим именем команды, если введённая строка и имя команды совпадает, команда выполняется;
* Если вводится команда завершения программы, программа завершается;
* Работа основной функции осуществляется через цикл while, который проверяет какой флаг возвращает функция, обрабатывающая команду (true – продолжается выполнения программы, false – программа завершает своё выполнения).

int main()

{

while (AcceptCommand());

return 0;

}

Метод AcceptCommand отвечает за обработку вводимых команд и возвращает значение, от которого зависит дальнейшее выполнение программы.

bool AcceptCommand()

{

cout << ">> ";

string command;

getline(cin, command);

if ((command == "exit") || (command == "0"))

return false;

else

{

if (command == "help")

ShowHelp();

else if (command == "newPersons")

InputNewPersons();

else if (command == "newString")

InputNewString();

else if (command == "lineSearch")

DoLineSearch();

else if (command == "interpSearch")

DoInterpolationSearch();

else if (command == "searchSubstr")

FindSubstring();

else if (command == "printPersons")

PrintAllPersons();

else if (command == "printString")

PrintString();

else

cout << "The command does not exist!!!" << endl;

return true;

}

}

Для каждой функции, не принимающих аргументов и описанных выше, задано соответствующее имя в программе. Для удобства была добавлена команда help, которая выводит список всех команд, присутствующих в программе с кратким описанием их предназначения.

void ShowHelp()

{

cout << "Commands:" << endl;

cout << " - newPersons -> enter a list of persons;" << endl;

cout << " - newString -> enter a new string;" << endl;

cout << " - lineSearch -> linear search of persons;" << endl;

cout << " - interpSearch -> interpolation search of persons;" << endl;

cout << " - searchSubstr -> search substring in the string;" << endl;

cout << " - printPersons -> persons list output;" << endl;

cout << " - printString -> string output;" << endl << endl;

}

**Полный исходный код программы на языке программирования C++:**

#include <iostream>

#include <string>

#include "Windows.h"

using namespace std;

struct Person

{

string fullName;

int passportNumber;

string phoneNumber;

};

int \_personsCount;

Person\* \_persons;

string \_string;

void PrintPerson(Person person, int index)

{

cout << endl << "Person #" << index + 1 << ":\n";

cout << " - Full name: " << person.fullName << endl;

cout << " - Passport number: " << person.passportNumber << endl;

cout << " - Phone number: " << person.phoneNumber << endl << endl;

}

void PrintPersons(Person\* persons)

{

for (int i = 0; i < \_personsCount; i++)

PrintPerson(persons[i], i);

}

bool PrintCompare(Person\* persons, int index, int compareValue)

{

if (persons[index].passportNumber == compareValue)

{

cout << "Item \"" << compareValue << "\" found: "

<< "index = " << index + 1 << ";\n";

PrintPerson(persons[index], index);

return true;

}

else

return false;

}

void SelectionSort(Person\*& persons)

{

int j = 0;

Person temp;

for (int i = 0; i < \_personsCount; i++)

{

j = i;

for (int k = i; k < \_personsCount; k++)

if (persons[j].passportNumber > persons[k].passportNumber)

j = k;

temp = persons[i];

persons[i] = persons[j];

persons[j] = temp;

}

}

void InterpolFindPeople(Person\* persons, int searchKey)

{

bool itemFound = false;

int

middle,

left = 0,

right = \_personsCount,

startIndex = -1;

while ((persons[left].passportNumber <= searchKey) && (persons[right].passportNumber >= searchKey))

{

middle = left + (searchKey - persons[left].passportNumber) \* (right - left) / (persons[right].passportNumber - persons[left].passportNumber);

if (persons[middle].passportNumber < searchKey)

left = middle + 1;

else if (persons[middle].passportNumber > searchKey)

right = middle - 1;

else

{

startIndex = middle;

itemFound = true;

break;

}

}

if (!itemFound)

cout << "The list has no items with the key;" << endl;

else

{

for (int i = startIndex; i < \_personsCount; i++)

{

if (!PrintCompare(persons, i, searchKey))

break;

}

for (int i = startIndex - 1; i >= 0; i--)

{

if (!PrintCompare(persons, i, searchKey))

break;

}

}

}

void LinearFindPeople(Person\* persons, int searchKey)

{

bool itemFound = false;

for (int i = 0; i < \_personsCount; i++)

if (persons[i].passportNumber == searchKey)

{

itemFound = true;

cout << "Item \"" << searchKey << "\" found: "

<< "index = " << i + 1 << ";\n";

PrintPerson(persons[i], i);

}

if (!itemFound)

cout << "The list has no items with the key;" << endl;

}

void FindSubstring(string& baseString, string& substring)

{

string foundSubstring = "";

int index = 0;

for (int i = 0; i < baseString.length(); i++)

{

if (baseString[i] == substring[index])

{

foundSubstring += baseString[i];

if (index == substring.length() - 1)

break;

index++;

}

else

{

foundSubstring = "";

index = 0;

}

}

if (foundSubstring == substring)

cout << "Substring \"" << substring << "\" has been found!" << endl;

else

cout << "The string doesn't contain \"" << substring << "\"!\n";

}

int ReadNaturalNum()

{

int x;

do cin >> x; while (x <= 0);

return x;

}

void ShowHelp()

{

cout << "Commands:" << endl;

cout << " - newPersons -> enter a list of persons;" << endl;

cout << " - newString -> enter a new string;" << endl;

cout << " - lineSearch -> linear search of persons;" << endl;

cout << " - interpSearch -> interpolation search of persons;" << endl;

cout << " - searchSubstr -> search substring in the string;" << endl;

cout << " - printPersons -> persons list output;" << endl;

cout << " - printString -> string output;" << endl << endl;

}

void InputNewPersons()

{

cout << "Enter the count of persons: ";

\_personsCount = ReadNaturalNum();

\_persons = new Person[\_personsCount];

for (int i = 0; i < \_personsCount; i++)

{

cout << endl << "Full name: ";

cin.ignore();

getline(cin, \_persons[i].fullName);

cout << "Passport number: ";

cin >> \_persons[i].passportNumber;

cout << "Phone number: ";

cin.ignore();

getline(cin, \_persons[i].phoneNumber);

}

cout << endl;

}

void InputNewString()

{

getline(cin, \_string);

}

void DoLineSearch()

{

cout << "Enter the key of searching (passport number): ";

int key = ReadNaturalNum();

LinearFindPeople(\_persons, key);

}

void DoInterpolationSearch()

{

cout << "Enter the key of searching (passport number): ";

int key = ReadNaturalNum();

SelectionSort(\_persons);

InterpolFindPeople(\_persons, key);

}

void FindSubstring()

{

cout << "Enter the substring, which you want to find: ";

string subString;

getline(cin, subString);

FindSubstring(\_string, subString);

}

void PrintAllPersons()

{

PrintPersons(\_persons);

}

void PrintString()

{

cout << \_string << endl;

}

bool AcceptCommand()

{

cout << ">> ";

string command;

getline(cin, command);

if ((command == "exit") || (command == "0"))

return false;

else

{

if (command == "help")

ShowHelp();

else if (command == "newPersons")

InputNewPersons();

else if (command == "newString")

InputNewString();

else if (command == "lineSearch")

DoLineSearch();

else if (command == "interpSearch")

DoInterpolationSearch();

else if (command == "searchSubstr")

FindSubstring();

else if (command == "printPersons")

PrintAllPersons();

else if (command == "printString")

PrintString();

else

cout << "The command does not exist!!!" << endl;

return true;

}

}

int main()

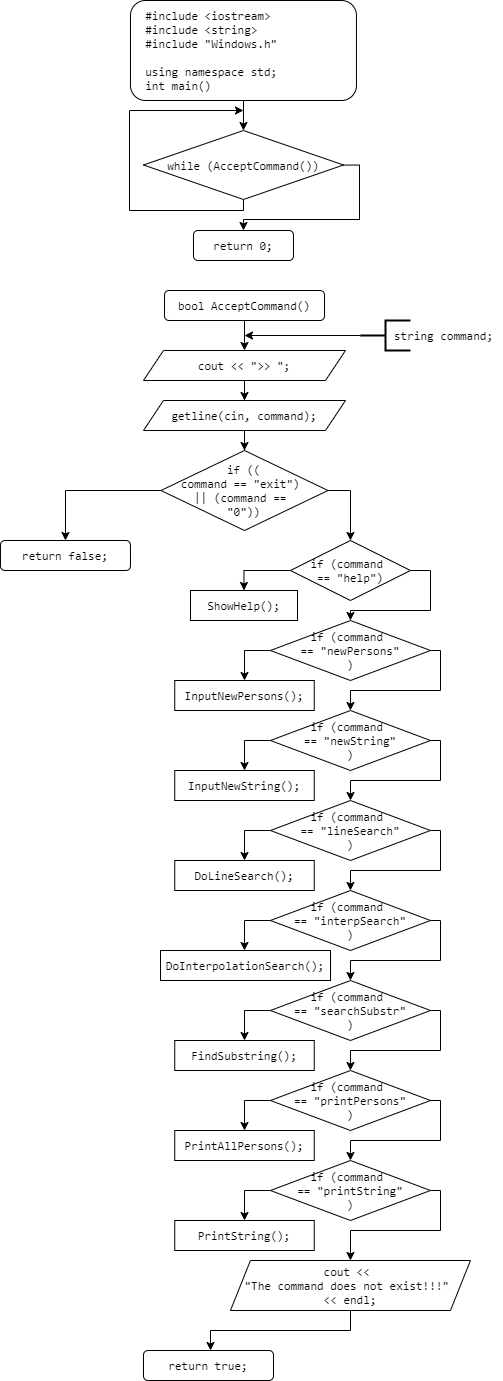
{

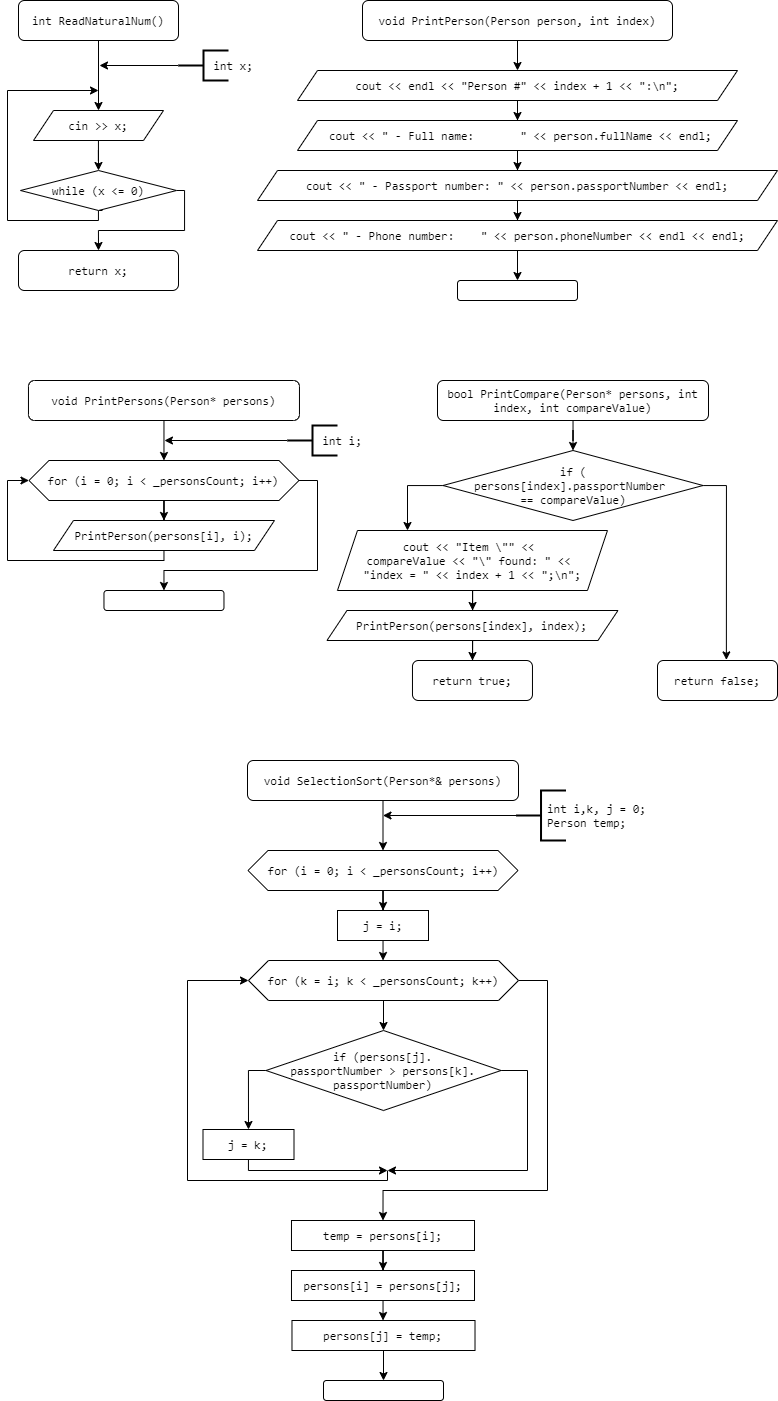
while (AcceptCommand());

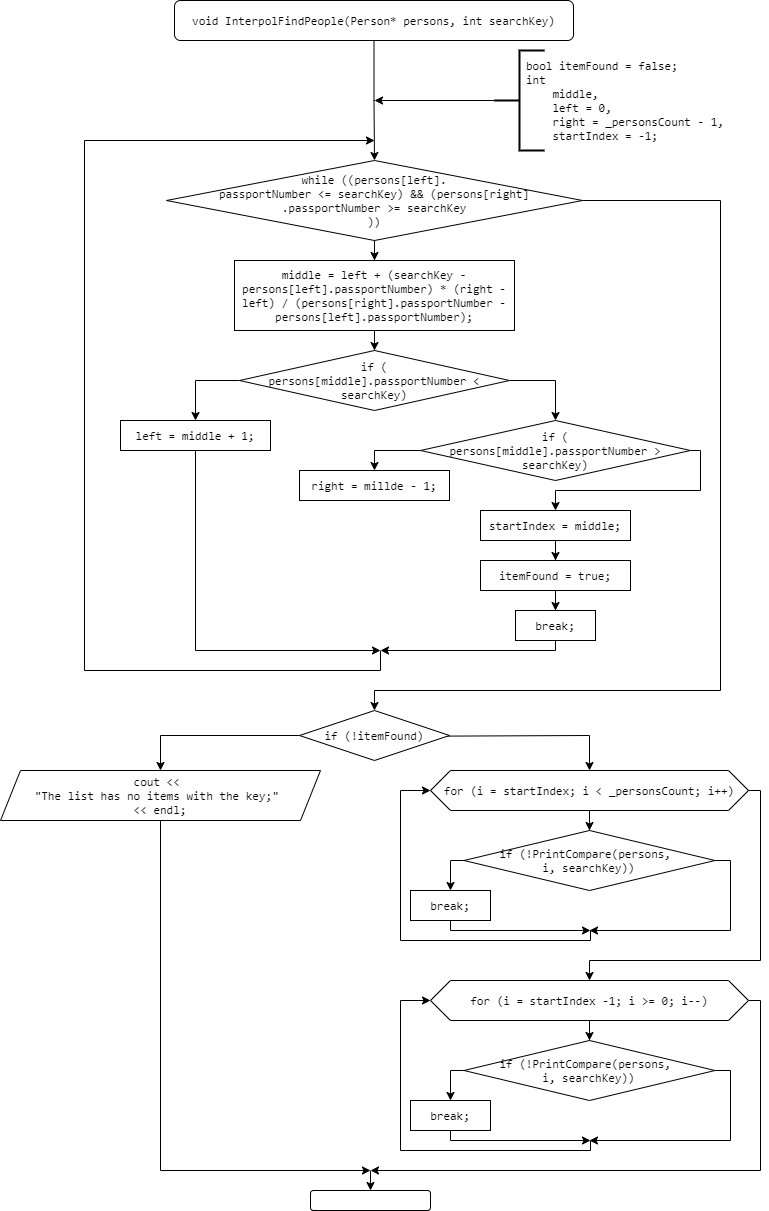
return 0;

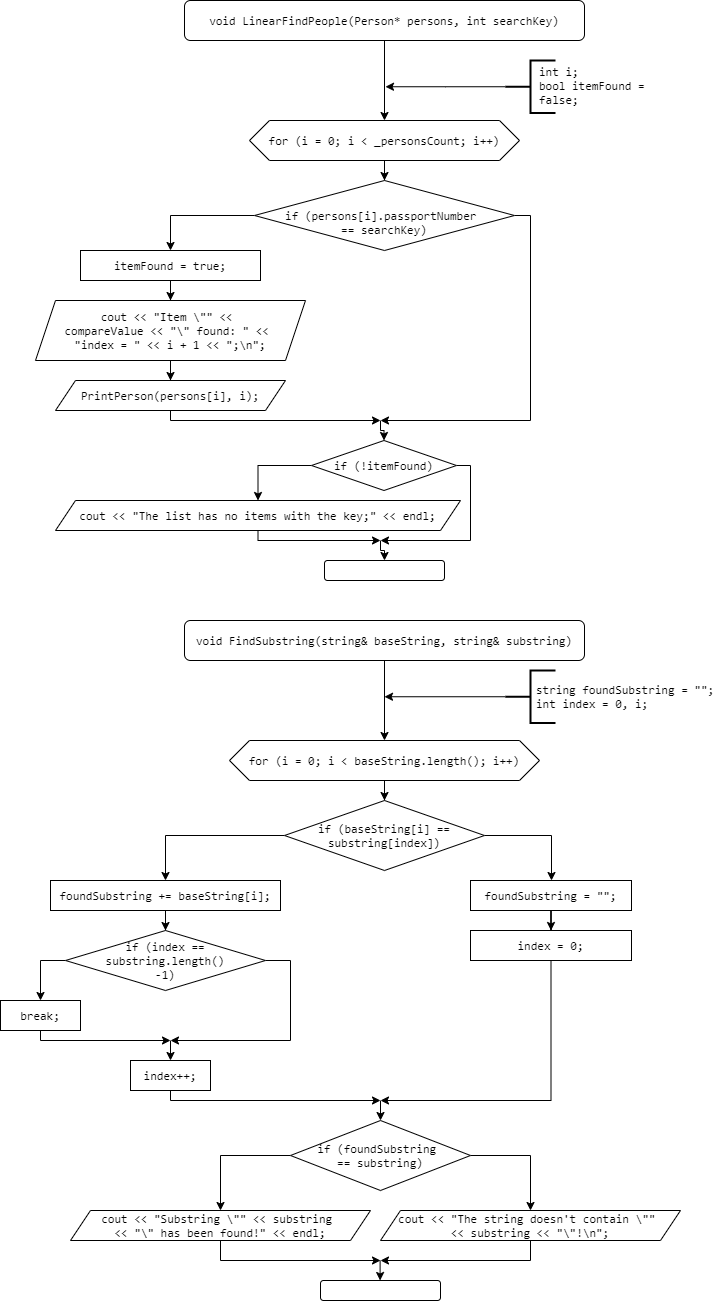
}

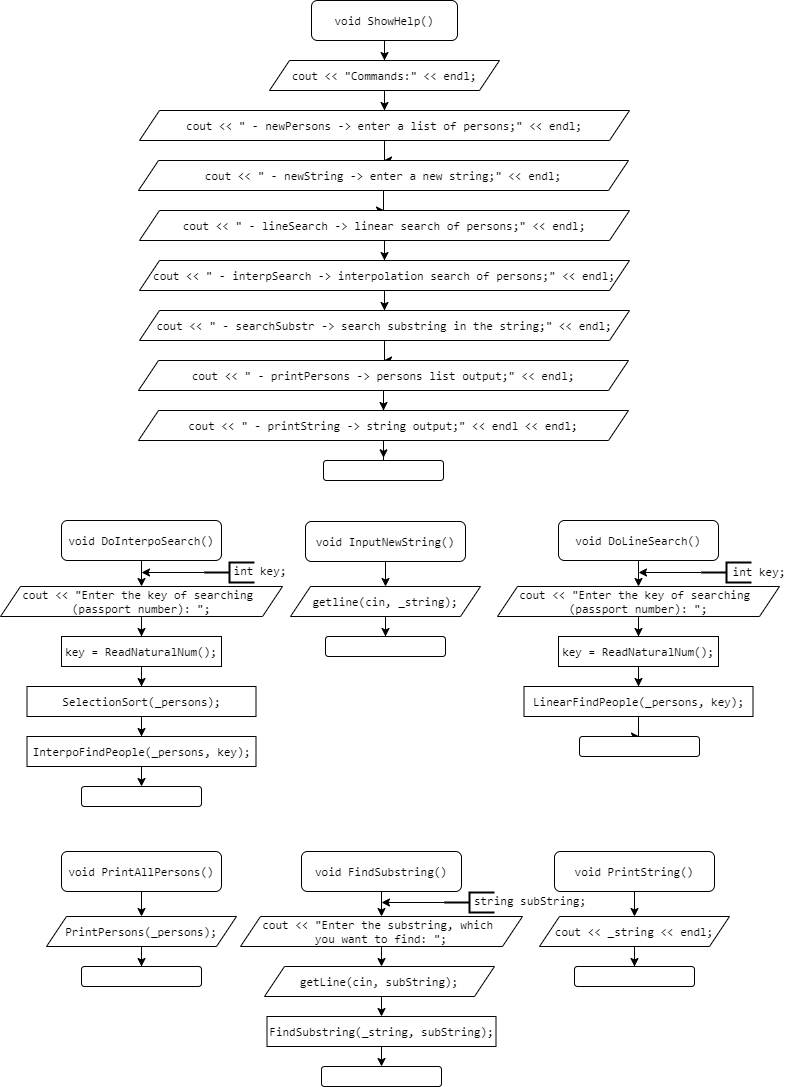
**Блок-схема** алгоритма программы:











**Скриншоты** выполненной программы:

