**«Пермский национальный исследовательский  
политехнический университет»**

**О Т Ч Ё Т**

**по лабораторной работе №17**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы быстрого поиска. Хеширование

Вариант 5

Выполнил работу

студент группы ИВТ-20-2б

Нефедов Л.В.

Проверила

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

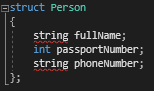
**Постановка задачи**

Реализовать на языке C++ программу, которая реализует следующий функционал:

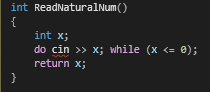
* Работа со структурами, заданными в варианте;
* Линейный поиск в структурах;
* Интерполяционный поиск в структурах;
* Линейный поиск подстроки в строке;
* Консольное меню для взаимодействия с программой;

**Анализ задачи**

1. Для решения задачи необходимо…
   1. Создать структуру Person для хранения данных различных типов в программе для удобства обращения к ним;

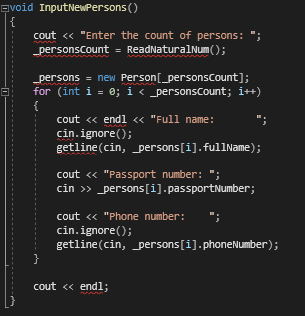


* 1. Длина массива и номер паспорта являются натуральными числами. Соответственно, при вводе целых чисел необходимо производить проверку на натуральность (если число больше нуля, то оно натуральное). Для ввода натуральных чисел была реализована функция ReadNaturalNum, которая решает данную задачу (если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

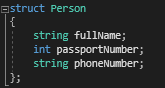


1.3 Для того, чтобы организовать ввод данных о новых людях, была реализована функция InputNewPersons. Функция просит ввести количество элементов нового массива и заполнить данных во всех элементах.

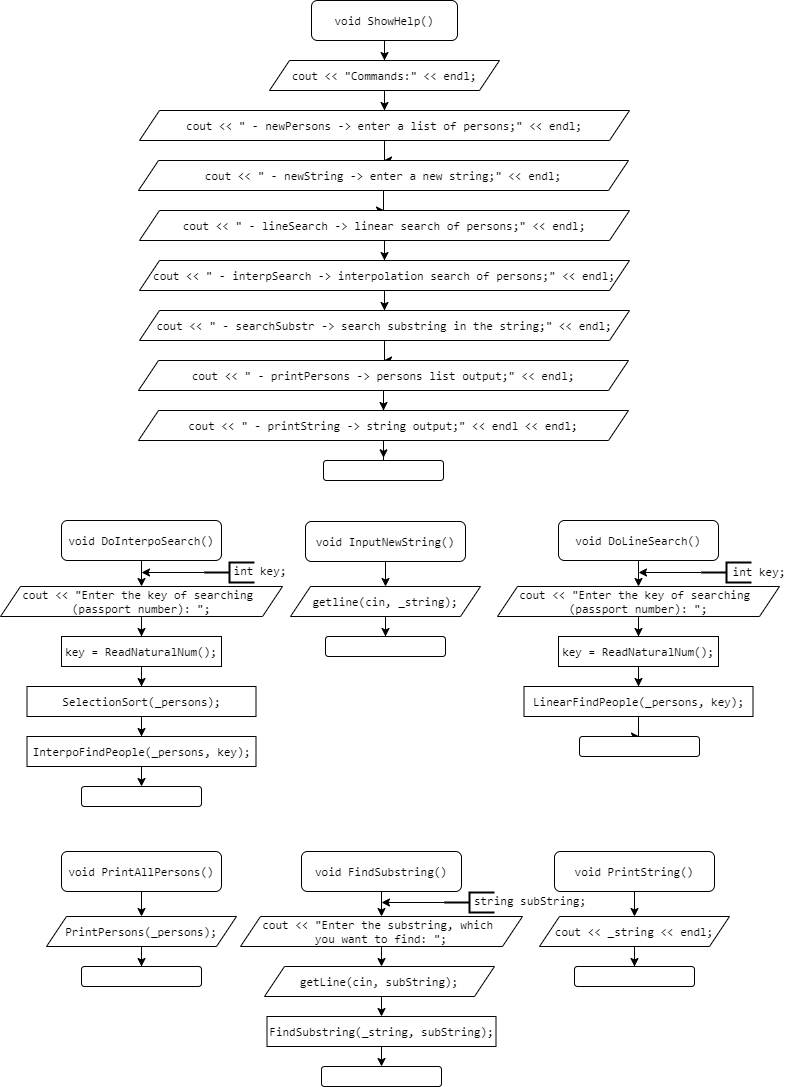
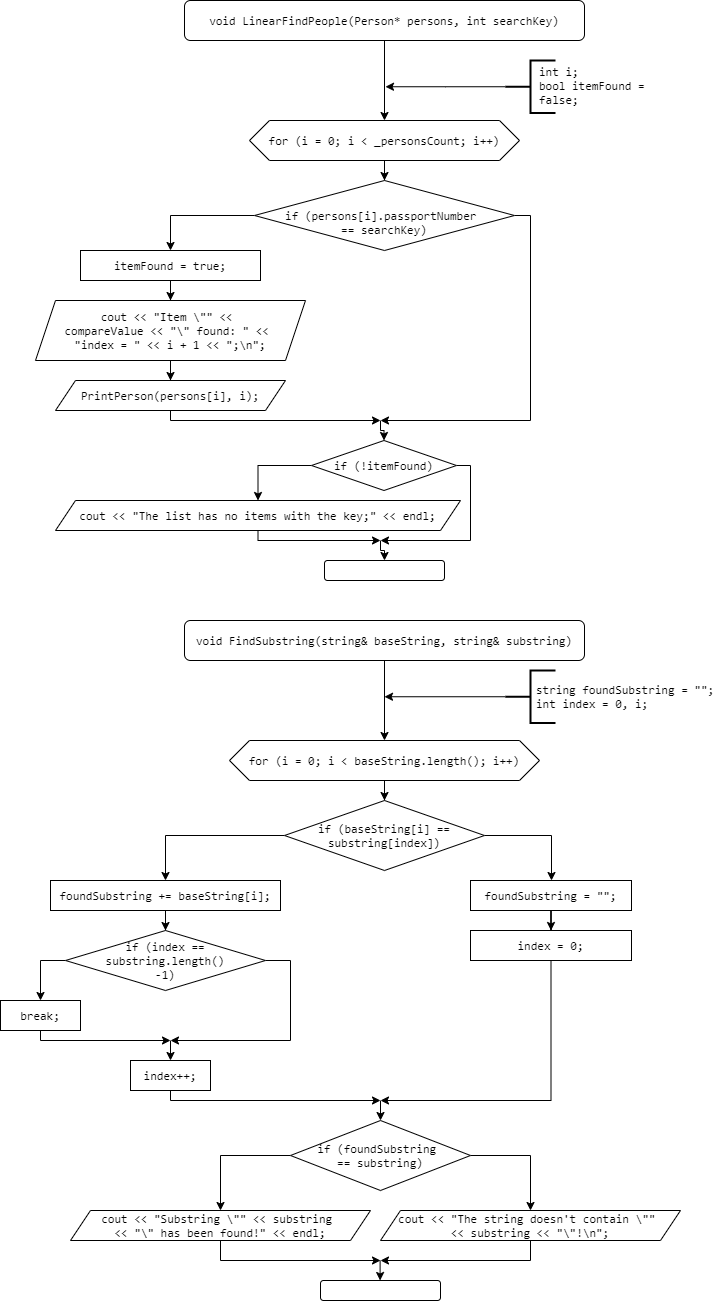
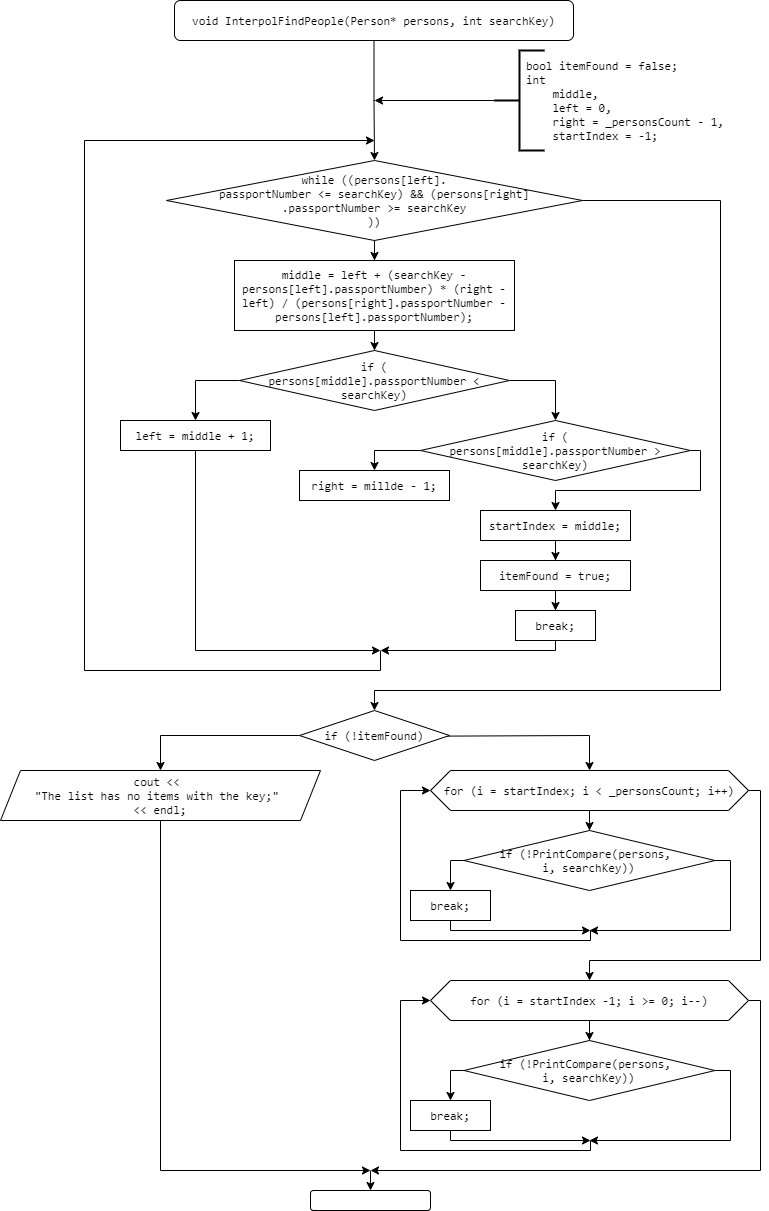
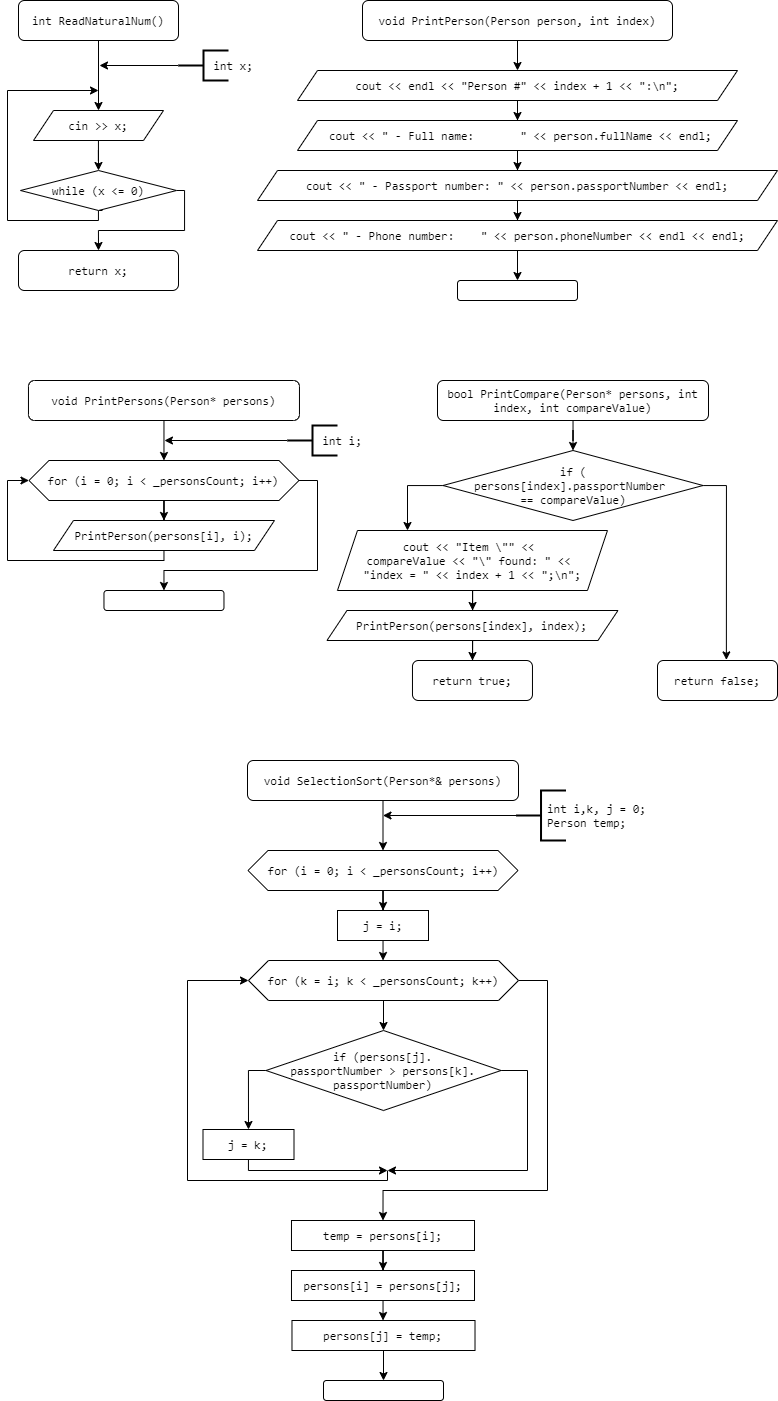
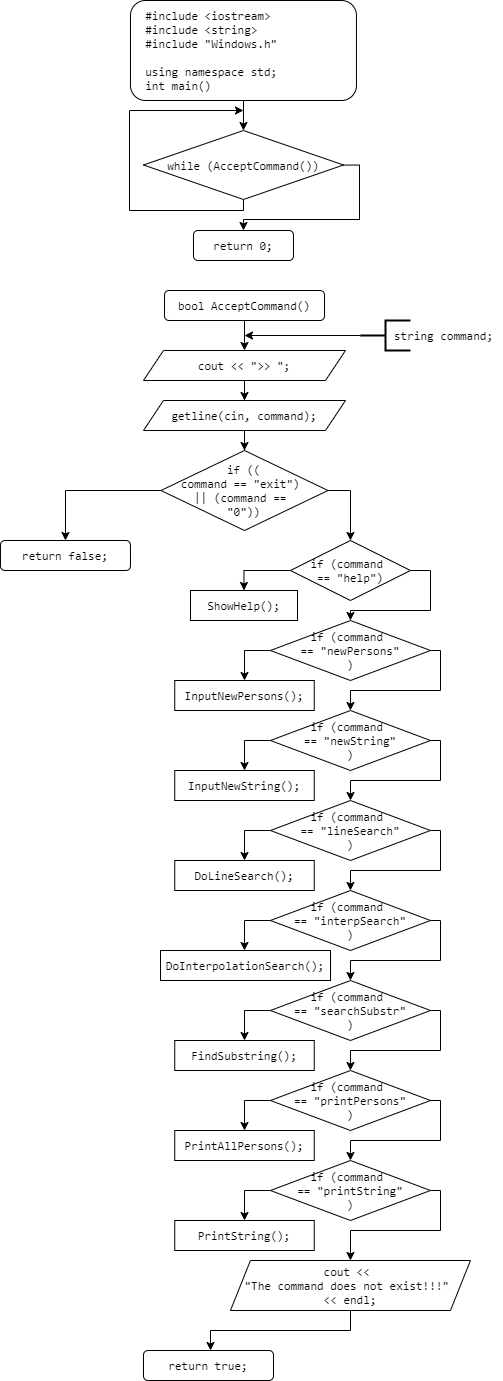
Чтобы вводить строки с пробелами, используется функция getline, которая принимает строку на ввод пока не будет нажат enter.



1. В программе были использованы следующие типы данных:



**Блок—схема программы**



**Решение**

#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;

}

**Скриншоты**

