МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет  
по практическим работам  
по дисциплине «Основы алгоритмизации и программирования»

Выполнил Проверил

студент группы ГИ-11 ст. преп. каф. ИУСиТ

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Содержание

[Практическая работа №14 Алгоритмы поиска 3](#_Toc35445566)

[Задания 3](#_Toc35445567)

[Контрольные вопросы 13](#_Toc35445568)

# Практическая работа №14 Алгоритмы поиска

### **Цель**

### Разработать алгоритмы поиска.

## Задания

### Условие

Разработать проект, включающий главную функцию, содержащую меню, и функции-подпрограммы:

1. Ввода или генерации одномерного массива. Количество элементов - не менее 10. Диапазон - на выбор. Вариативность оценивается (тип и диапазон значений массива, количество элементов).
2. Функция, позволяющая поменять местами значения двух чисел (не включается в меню) swap(a, b)
3. Функция сортировки (на выбор)
4. Функция последовательного прямого поиска
5. Функция бинарного поиска
6. Функция поиска методом транспозиции
7. Функция поиска с перемещением в начало
8. Функция индексно-последовательного поиска
9. Функция вывода одномерного массива

**Программный код**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <limits.h>

#include <locale.h>

void bubbleSort(int Mass[10], int size)

{for (int i=0; i<10-1; i++)

{

for (int j=(size-1); j>i; j--)

{

if (Mass[j-1]>Mass[j])

{

swap(&Mass[j], &Mass[j-1]);

}

}

}

}

int simpleSearch (int Mass[10], int num[10], int size, int value)

{

int j=0;

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

{

if (Mass[i]==value)

{

num[j++]=i;

}

}

return j;

}

int binarySearch(int Mass[10], int left, int right, int value)

{

if (left<right)

{

int medium=(left+right)/2;

if (Mass[medium]<value)

{

return binarySearch(Mass, medium+1, right, value);}

else if (Mass[medium]>value)

{

return binarySearch(Mass, left, medium-1, value);}

else

{return medium;}

} else if (left==right && Mass[left]==value)

{return left;}

else {return -1;}

}

void out(int Mass[10], int size)

{

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

{

printf("%d ", Mass[i]);

}

printf("\n");}

int transposSearch(int Mass[10], int size, int value)

{

if (Mass[0]==value){return 0;}

else

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

{if (Mass[i]==value){

swap(&Mass[i],&Mass[i-1]);

return i;}}}

return -1;

}

int searchNear(int Mass[10], int size, int pos)

{

if (pos!=-1)

{

int j=1;

int temp=pos-1;

while (temp>=0 && Mass[temp]==Mass[pos]){

temp--;

j++;}

temp=pos+1;

while (temp<size && Mass[temp]==Mass[pos]){

temp++;

j++;}

return j;

} else {return 0;}

}

int startSearch(int Mass[10], int size, int value)

{

for (int i=0; i<size;i++){if (Mass[i]==value){

swap(&Mass[i],&Mass[0]);

return i;}}

return -1;}

int indexSearch(int Mass[10], int size, int value)

{

int indexTable[size/3], i;

for (i=2; i<size; i+=3)

{indexTable[i/3]=Mass[i];}

for (i=0; i<size/3; i++){

if(indexTable[i]>value){

for(int j=i\*3; j<(i+1)\*3; j++)

{if (Mass[j]==value){return j;}}

return -1;}

else if (indexTable[i]==value)

{return i\*3+2;}}

for (i=(i+1)\*3; i<size; i++)

{if (indexTable[i]==value){return i;}}

return -1;

}

void generate(int Mass[10], int size)

{

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

{

Mass[i]=rand()%25-10;

}

}

void swap(int \*i, int \*j)

{int tmp;

tmp=\*i;

\*i=\*j;

\*j=tmp;

}

int main()

{ int size=10, Mass[10], value, num[10], number;

setlocale(LC\_ALL, "");

generate(Mass, size);

bubbleSort(Mass, size);

out(Mass, size);

printf("Введите номер задания: ");

scanf("%d", &number);

switch (number){

case 1:

printf("Значение элемента:");

scanf("%d", &value);

printf("simpleSearch:\n");

number=simpleSearch(Mass, num, size, value);

out(num, number);

break;

case 2:

printf("Значение элемента:");

scanf("%d", &value);

printf("binarySearch\n");

printf("Индекс элемента:1%d\n", number=binarySearch(Mass, 0, size-1, value));

break;

case 3:

printf("Значение элемента:");

scanf("%d", &value);

printf("transposSearch\n");

printf("Индекс элемента:%d\n", number=transposSearch(Mass, size, value));

break;

case 4:

printf("Значение элемента:");

scanf("%d", &value);

printf("startSearch\n");

printf("Индекс элемента:%d\n", number=startSearch(Mass, size, value));

break;

case 5:

printf("Значение элемента:");

scanf("%d", &value);

printf("indexSearch\n");

printf("Индекс элемента:%d\n", number=indexSearch(Mass, size, value));

break;

}

return 0;

}

### **Задание 1.1**

### Условие

Ввода или генерации одномерного массива. Количество элементов - не менее 10. Диапазон - на выбор. Вариативность оценивается (тип и диапазон значений массива, количество элементов).

### Листинг

void out(int Mass[10], int size)

{

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

{

printf("%d ", Mass[i]);

}

printf("\n");}

### **Задание 1.2**

### Условие

Функция, позволяющая поменять местами значения двух чисел (не включается в меню) swap(a, b).

### Листинг

void swap(int \*i, int \*j)

{int tmp;

tmp=\*i;

\*i=\*j;

\*j=tmp;}

### **Задание 1.3**

### Условие

Функция сортировки (на выбор).

### Листинг

void bubbleSort(int Mass[10], int size)

{for (int i=0; i<10-1; i++)

{

for (int j=(size-1); j>i; j--)

{

if (Mass[j-1]>Mass[j])

{

swap(&Mass[j], &Mass[j-1]);

}

}

}

}

### **Задание 1.4**

### Условие

Функция последовательного прямого поиска

### Листинг

int simpleSearch (int Mass[10], int num[10], int size, int value)

{

int j=0;

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

{

if (Mass[i]==value)

{

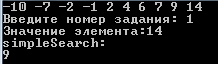
num[j++]=i;

}

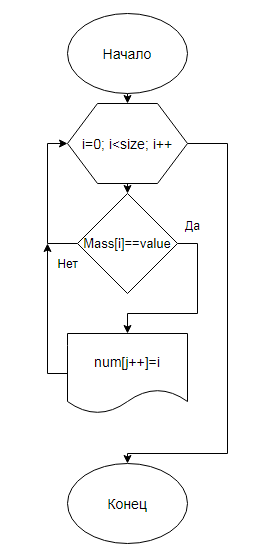
}

return j;

}

  
Рисунок 1 – Решение задания 1.4

**Блок-схема:**



### **Задание 1.5**

### Условие

Функция бинарного поиска

### Листинг

int binarySearch(int Mass[10], int left, int right, int value)

{

if (left<right)

{

int medium=(left+right)/2;

if (Mass[medium]<value)

{

return binarySearch(Mass, medium+1, right, value);}

else if (Mass[medium]>value)

{

return binarySearch(Mass, left, medium-1, value);}

else

{return medium;}

} else if (left==right && Mass[left]==value)

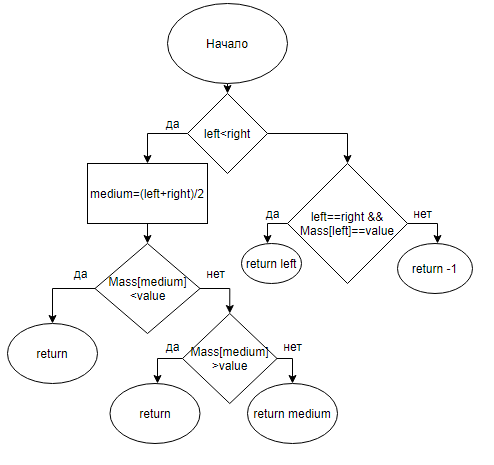
{return left;}

else {return -1;}

}

  
Рисунок 1 – Решение задания 1.5

**Блок-схема:**



### **Задание 1.6**

### Условие

Функция поиска методом транспозиции

### Листинг

int transposSearch(int Mass[10], int size, int value)

{

if (Mass[0]==value){return 0;}

else

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

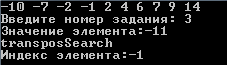
{if (Mass[i]==value){

swap(&Mass[i],&Mass[i-1]);

return i;}}}

return -1;

}

  
Рисунок 1 – Решение задания 1.6

### **Задание 1.7**

### Условие

Функция поиска с перемещением в начало

### Листинг

int searchNear(int Mass[10], int size, int pos)

{

if (pos!=-1)

{

int j=1;

int temp=pos-1;

while (temp>=0 && Mass[temp]==Mass[pos]){

temp--;

j++;}

temp=pos+1;

while (temp<size && Mass[temp]==Mass[pos]){

temp++;

j++;}

return j;

} else {return 0;}

}

int startSearch(int Mass[10], int size, int value)

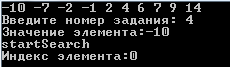
{

for (int i=0; i<size;i++){if (Mass[i]==value){

swap(&Mass[i],&Mass[0]);

return i;}}

return -1;}

  
Рисунок 1 – Решение задания 1.7

### **Задание 1.8**

### Условие

Функция индексно-последовательного поиска

### Листинг

int indexSearch(int Mass[10], int size, int value)

{

int indexTable[size/3], i;

for (i=2; i<size; i+=3)

{indexTable[i/3]=Mass[i];}

for (i=0; i<size/3; i++){

if(indexTable[i]>value){

for(int j=i\*3; j<(i+1)\*3; j++)

{if (Mass[j]==value){return j;}}

return -1;}

else if (indexTable[i]==value)

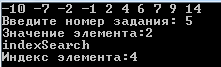
{return i\*3+2;}}

for (i=(i+1)\*3; i<size; i++)

{if (indexTable[i]==value){return i;}}

return -1;

}

  
Рисунок 1 – Решение задания 1.8

### **Задание 1.9**

### Условие

Функция вывода одномерного массива

### Листинг

void out(int Mass[10], int size)

{

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

{

printf("%d ", Mass[i]);

}

printf("\n");}

## Контрольные вопросы

1. Описание алгоритма последовательного прямого поиска

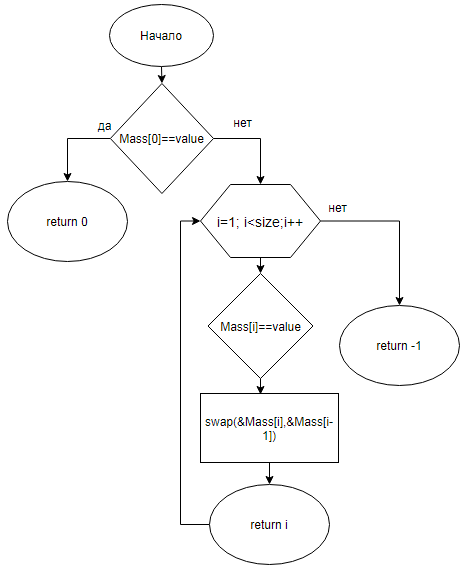
Идём по порядку по массиву, находим нужный элемент.

1. Описание алгоритма бинарного поиска

Двоичный поиск заключается в том, что на каждом шаге множество объектов делится на две части и в работе остаётся та часть множества, где находится искомый объект..

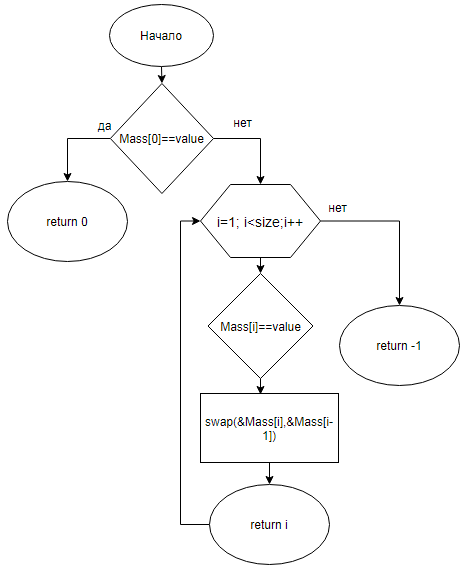
1. На 7-10. Описание алгоритма последовательного поиска с транспозицией. Блок-схема алгоритма

Последовательный поиск с транспозицией это улучшенная версия последовательного поиска. Каждый запрос к элементу сопровождается сменой мест этого и предшествующего элемента. В итоге наиболее часто используемые элементы постепенно перемещаются в начало массива. И при последующем обращении к ним, эти записи находятся почти сразу.



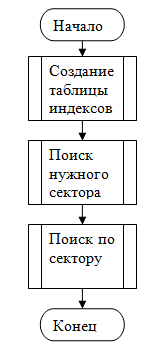
1. На 7-10. Описание алгоритма последовательного поиска с перемещением в начало. Блок-схема алгоритма

В этом методе каждый запрос к элементу сопровождается его перемещением в начало массива. В итоге в начале массива оказывается элемент, используемый в последний раз.



1. На 7-10. Описание алгоритма индексно-последовательного поиска. Блок-схема алгоритма

Достоинство алгоритма индексно-последовательного поиска заключается в том, что сокращается время поиска, так как последовательный поиск первоначально ведется в индексной таблице, имеющей меньший размер, чем основная таблица. Когда найден правильный индекс, второй последовательный поиск выполняется по небольшой части записей основной таблицы.



Вывод по работе

Разработал алгоритмы поиска.