Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Методы поиска(Линейный, интерполяционный, Бойера-Мура, Кнута-Морриса-Пратта, Шелла)

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Мухаров Тимур Рустамович

Проверил

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Г. Пермь-2023

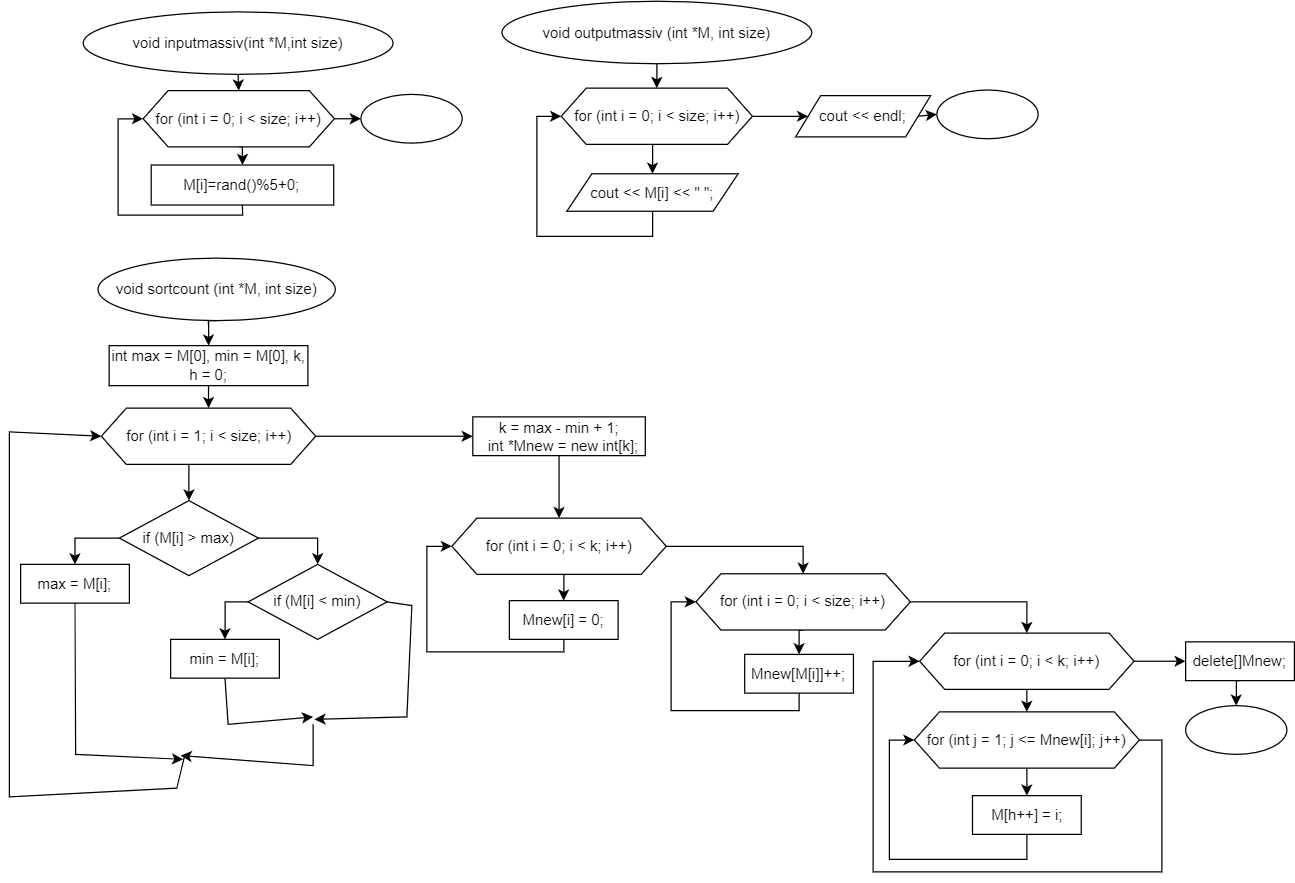
**Постановка задачи**

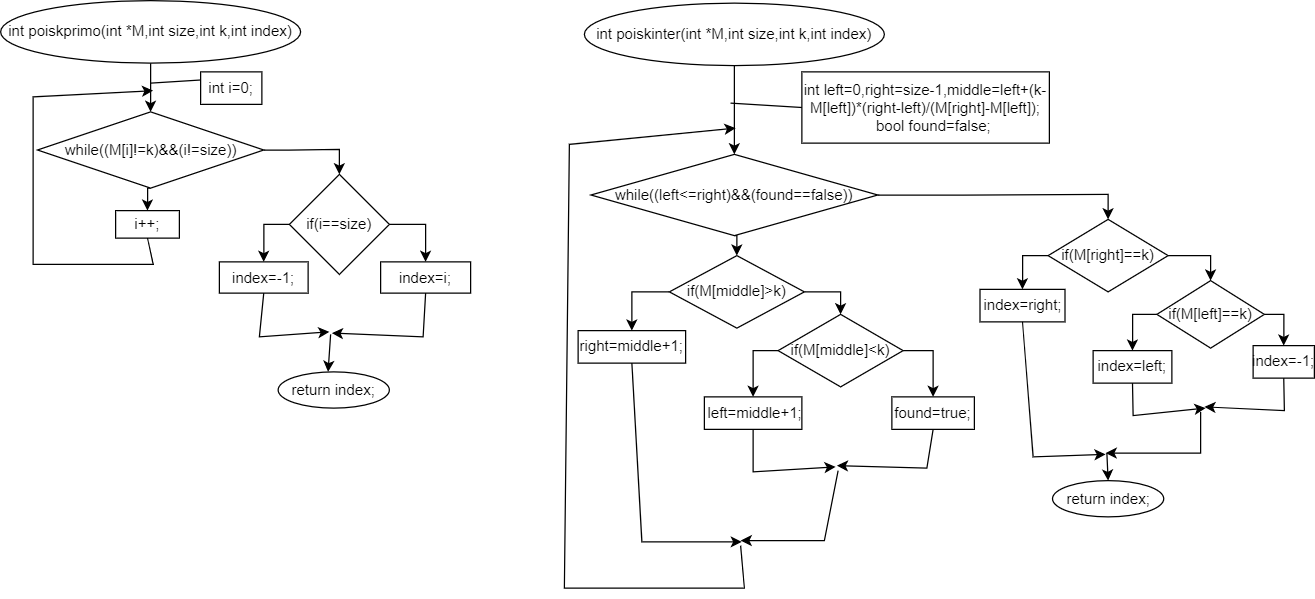
Опробовать все методы поиска.

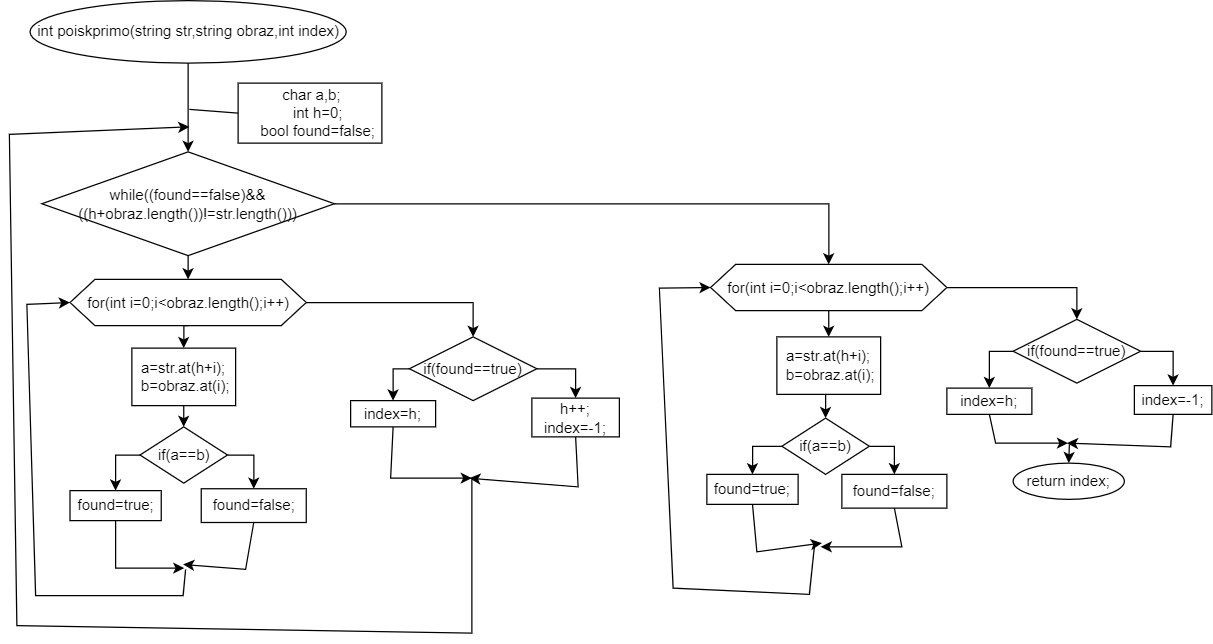
**Анализ решения**

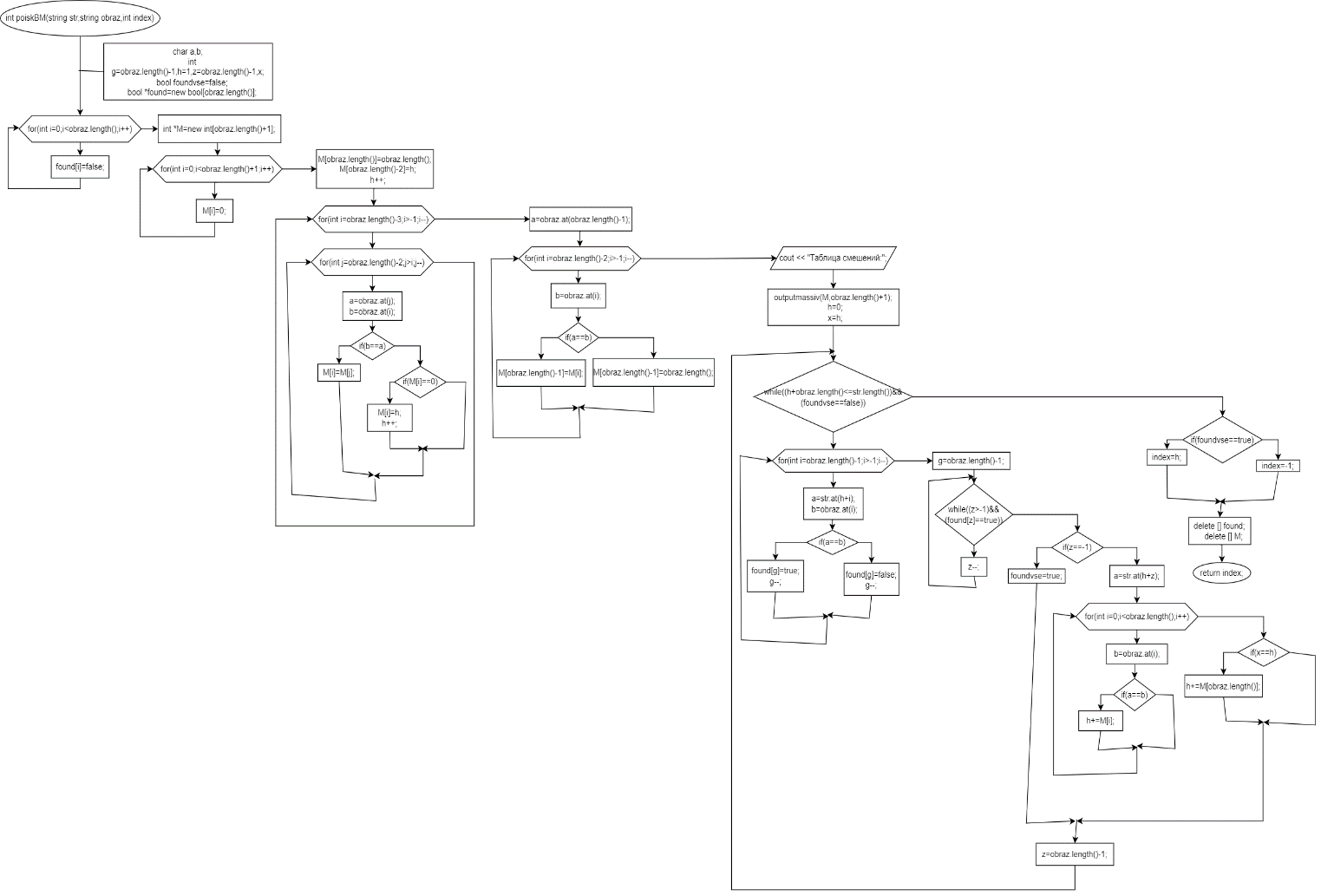
Каждый метод поиска засунуть в отдельную функцию.

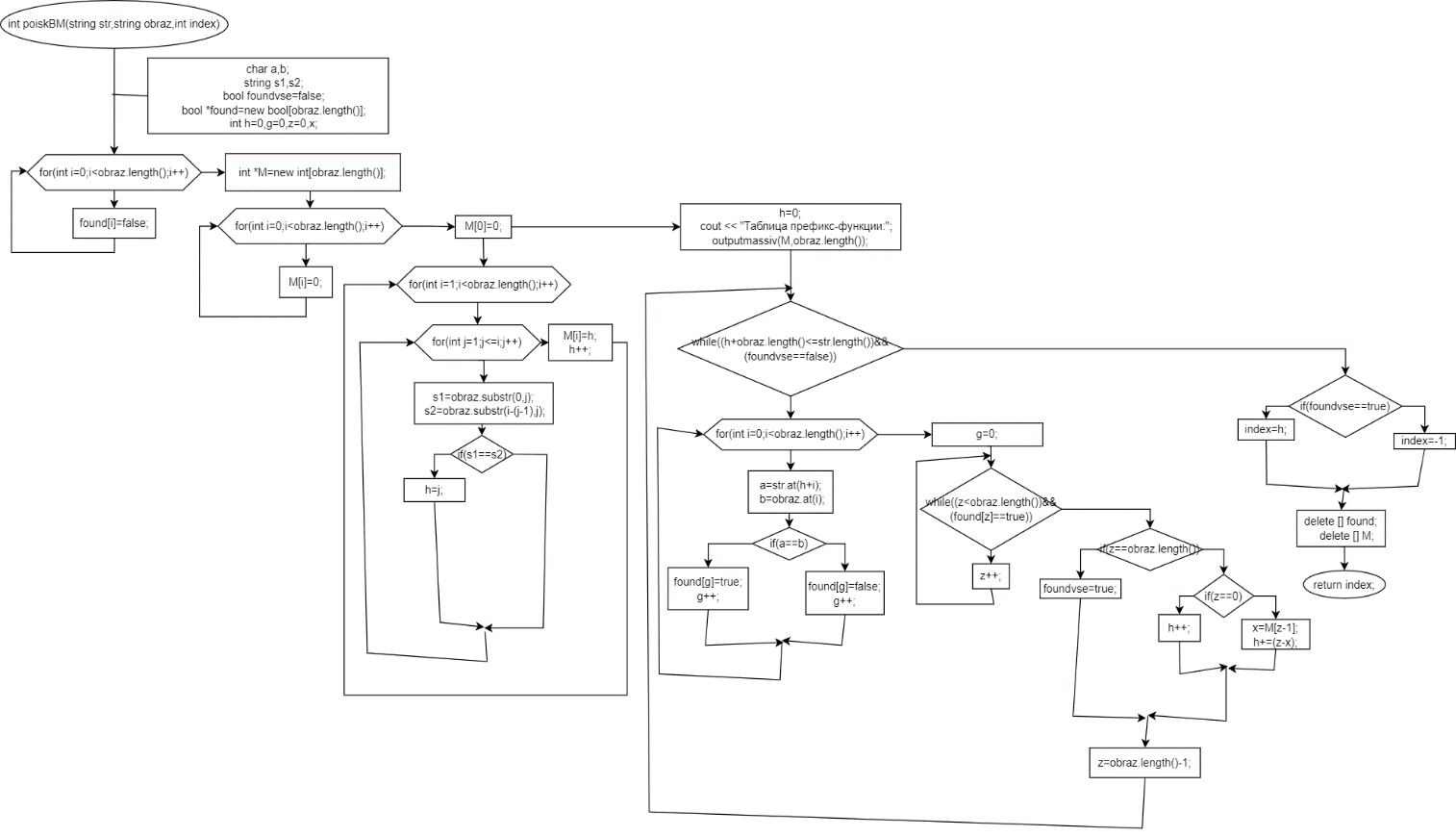
**Алгоритм программы**

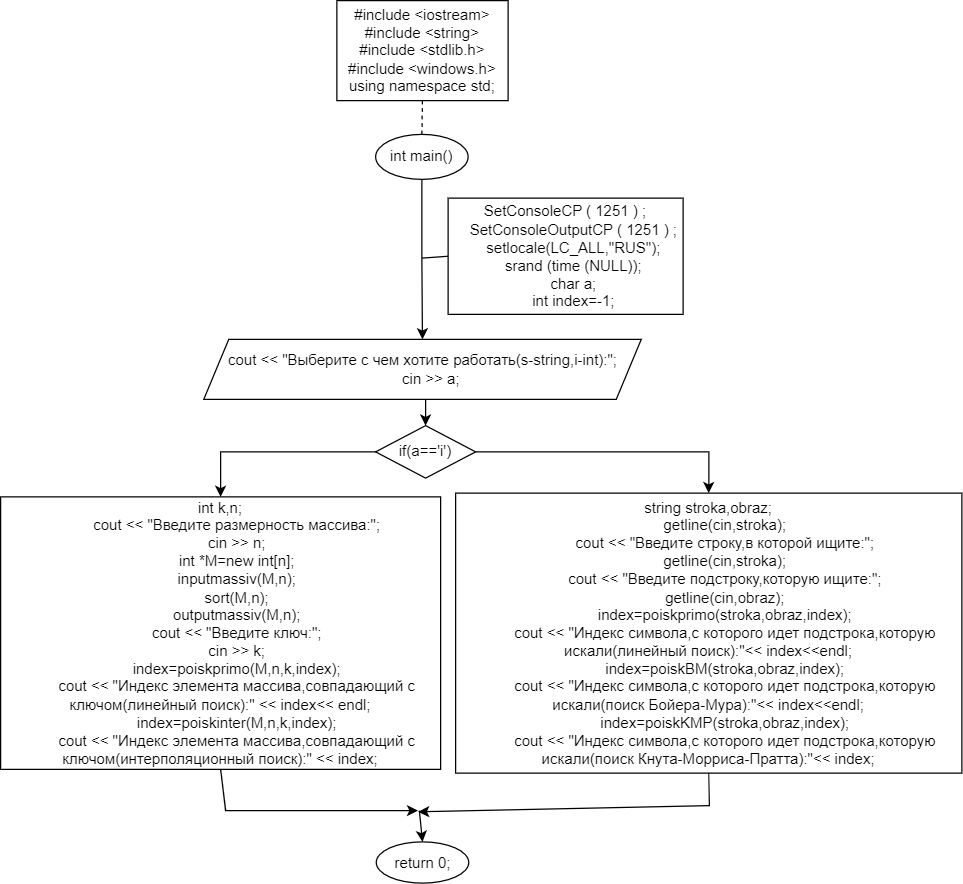


****

****

****

****

****

Код программы:

#include <iostream>

#include <string>

#include <stdlib.h>

#include <windows.h>

using namespace std;

void inputmassiv(int \*M,int size)

{

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

{

M[i]=rand()%5+0;

}

}

void outputmassiv (int \*M, int size)

{

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

{

cout << M[i] << " ";

}

cout << endl;

}

void sort (int \*M, int size)

{

int max = M[0], min = M[0], k, h = 0;

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

{

if (M[i] > max)

{

max = M[i];

}

else

{

if (M[i] < min)

{

min = M[i];

}

}

}

k = max - min + 1;

int \*Mnew = new int[k];

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

{

Mnew[i] = 0;

}

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

{

Mnew[M[i]]++;

}

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

{

for (int j = 1; j <= Mnew[i]; j++)

{

M[h++] = i;

}

}

delete[]Mnew;

}

int poiskprimo(int \*M,int size,int k,int index)

{

int i=0;

while((M[i]!=k)&&(i!=size))

{

i++;

}

if(i==size)

{

index=-1;

}

else

{

index=i;

}

return index;

}

int poiskinter(int \*M,int size,int k,int index)

{

int left=0,right=size-1,middle=left+(k-M[left])\*(right-left)/(M[right]-M[left]);

bool found=false;

while((left<=right)&&(found==false))

{

if(M[middle]>k)

{

right=middle+1;

}

else

{

if(middle<k)

{

left=middle+1;

}

else

{

found=true;

}

}

}

if(M[right]==k)

{

index=right;

}

else

{

if(M[left]==k)

{

index=left;

}

else

{

index=-1;

}

}

return index;

}

int poiskprimo(string str,string obraz,int index)

{

char a,b;

int h=0;

bool found=false;

while((found==false)&&((h+obraz.length())!=str.length()))

{

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

{

a=str.at(h+i);

b=obraz.at(i);

if(a==b)

{

found=true;

}

else

{

found=false;

}

}

if(found==true)

{

index=h;

}

else

{

h++;

index=-1;

}

}

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

{

a=str.at(h+i);

b=obraz.at(i);

if(a==b)

{

found=true;

}

else

{

found=false;

}

}

if(found==true)

{

index=h;

}

else

{

index=-1;

}

return index;

}

int poiskBM(string str,string obraz,int index)

{

char a,b;

int g=obraz.length()-1,h=1,z=obraz.length()-1,x;

bool foundvse=false;

bool \*found=new bool[obraz.length()];

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

{

found[i]=false;

}

int \*M=new int[obraz.length()+1];

for(int i=0;i<obraz.length()+1;i++)

{

M[i]=0;

}

M[obraz.length()]=obraz.length();

M[obraz.length()-2]=h;

h++;

for(int i=obraz.length()-3;i>-1;i--)

{

for(int j=obraz.length()-2;j>i;j--)

{

a=obraz.at(j);

b=obraz.at(i);

if(b==a)

{

M[i]=M[j];

}

else

{

if(M[i]==0)

{

M[i]=h;

h++;

}

}

}

}

a=obraz.at(obraz.length()-1);

for(int i=obraz.length()-2;i>-1;i--)

{

b=obraz.at(i);

if(a==b)

{

M[obraz.length()-1]=M[i];

}

else

{

M[obraz.length()-1]=obraz.length();

}

}

cout << "Таблица смешений:";

outputmassiv(M,obraz.length()+1);

h=0;

x=h;

while((h+obraz.length()<=str.length())&&(foundvse==false))

{

for(int i=obraz.length()-1;i>-1;i--)

{

a=str.at(h+i);

b=obraz.at(i);

if(a==b)

{

found[g]=true;

g--;

}

else

{

found[g]=false;

g--;

}

}

g=obraz.length()-1;

while((z>-1)&&(found[z]==true))

{

z--;

}

if(z==-1)

{

foundvse=true;

}

else

{

a=str.at(h+z);

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

{

b=obraz.at(i);

if(a==b)

{

h+=M[i];

}

}

if(x==h)

{

h+=M[obraz.length()];

}

}

z=obraz.length()-1;

}

if(foundvse==true)

{

index=h;

}

else

{

index=-1;

}

delete [] found;

delete [] M;

return index;

}

int poiskKMP(string str,string obraz,int index)

{

char a,b;

string s1,s2;

bool foundvse=false;

bool \*found=new bool[obraz.length()];

int h=0,g=0,z=0,x;

int \*M=new int[obraz.length()];

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

{

M[i]=0;

}

M[0]=0;

for(int i=1;i<obraz.length();i++)

{

for(int j=1;j<=i;j++)

{

s1=obraz.substr(0,j);

s2=obraz.substr(i-(j-1),j);

if(s1==s2)

{

h=j;

}

}

M[i]=h;

h=0;

}

h=0;

cout << "Таблица префикс-функции:";

outputmassiv(M,obraz.length());

while((h+obraz.length()<=str.length())&&(foundvse==false))

{

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

{

a=str.at(h+i);

b=obraz.at(i);

if(a==b)

{

found[g]=true;

g++;

}

else

{

found[g]=false;

g++;

}

}

g=0;

while((z<obraz.length())&&(found[z]==true))

{

z++;

}

if(z==obraz.length())

{

foundvse=true;

}

else

{

if(z==0)

{

h++;

}

else

{

x=M[z-1];

h+=(z-x);

}

}

z=0;

}

if(foundvse==true)

{

index=h;

}

else

{

index=-1;

}

delete [] found;

delete [] M;

return index;

}

int main()

{

SetConsoleCP ( 1251 ) ;

SetConsoleOutputCP ( 1251 ) ;

setlocale(LC\_ALL,"RUS");

srand (time (NULL));

char a;

int index=-1;

cout << "Выберите с чем хотите работать(s-string,i-int):";

cin >> a;

if(a=='i')

{

int k,n;

cout << "Введите размерность массива:";

cin >> n;

int \*M=new int[n];

inputmassiv(M,n);

sort(M,n);

outputmassiv(M,n);

cout << "Введите ключ:";

cin >> k;

index=poiskprimo(M,n,k,index);

cout << "Индекс элемента массива,совпадающий с ключом(линейный поиск):" << index<< endl;

index=poiskinter(M,n,k,index);

cout << "Индекс элемента массива,совпадающий с ключом(интерполяционный поиск):" << index;

}

else

{

string stroka,obraz;

getline(cin,stroka);

cout << "Введите строку,в которой ищите:";

getline(cin,stroka);

cout << "Введите подстроку,которую ищите:";

getline(cin,obraz);

index=poiskprimo(stroka,obraz,index);

cout << "Индекс символа,с которого идет подстрока,которую искали(линейный поиск):"<< index<<endl;

index=poiskBM(stroka,obraz,index);

cout << "Индекс символа,с которого идет подстрока,которую искали(поиск Бойера-Мура):"<< index<<endl;

index=poiskKMP(stroka,obraz,index);

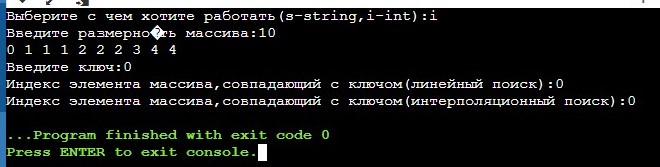
cout << "Индекс символа,с которого идет подстрока,которую искали(поиск Кнута-Морриса-Пратта):"<< index;

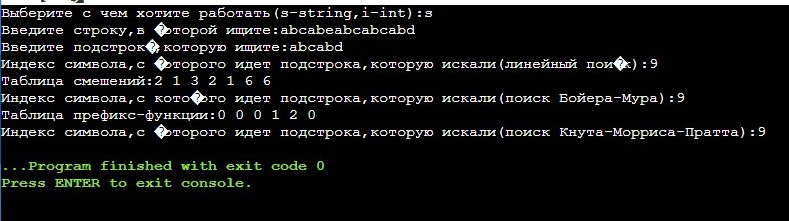
}

return 0;

}

**Результаты работы программы**





**Анализ результатов**

Как видно из результатов работы программы, диалог с пользователем хорошо настроен.