ТЕМА 7. Алгоритмы использующие линейные СВЯЗАННЫЕ СПИСКИ

7.1. Задача вычисления арифметических выражений

s:=(a+b)*(c+d)/f;

a+b **инфиксная форма**

+ab **префиксная форма**

ab+ **постфиксная форма**

Алгоритм преобразования выражения из инфиксной формы в форму обратной польской записи

Операции	)	(	+	–	*	/	*(возв. в ст.)*
Приоритет	**0**		**1**			**2**	**3**

7.2. Программа для вычисления арифметических выражений (с использованием стека)

```
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct tstk
 {
   char inf;     tstk *a;
};

struct tstkd
 {
   double inf;      tstkd *a;
};
```

```
tstk *AddStack(tstk *sp, char inf)
{   tstk *spt=new tstk;
    spt->inf = inf;
    spt->a = sp;
 return spt;  }

tstkd *AddStack(tstkd *sp, double inf)
{   tstkd *spt=new tstkd;
    spt->inf = inf;
    spt->a = sp;
 return spt;  }
```

```
tstk *ReadStackD(tstk *sp, char &inf)
{
 tstk *spt = sp;       inf= sp->inf;
   sp = sp->a;
  delete spt;
 return sp;  }
tstkd *ReadStackD(tstkd *sp, double &inf)
{
 tstkd *spt = sp;         inf= sp->inf;
   sp = sp->a;
  delete spt;
 return sp;  }
```

```
double masz[122];
char str[100],   strp[100];
```

```
int priority(char ch)  // Вычисление
                       //приоритета операций
{
 switch (ch)
 {
    case '(': case ')‘    : return 0;
    case '+': case '-‘    : return 1;
    case '*': case '/‘    : return 2;
    default: return -1;
 }
}
```

```
void AddPostFix(char *strin, char *strout)
{
tstk *sp=NULL;
int n = 0;
char ch, inf;
```

```
for (unsigned int i=0; i<strlen(strin); i++)
{
ch=strin[i];
// Если это операнд
if (ch >= 'A') { strout[n++]=ch; continue; }
// Если стек пуст или открыв.скобка
if (sp == NULL || ch == '(' )
{ sp=AddStack(sp,ch); continue; }
```

```
// Если закрывающая скобка
if ( ch == ')' )
{
    while (sp->inf != '(')
    {
      sp=ReadStackD(sp, (char) inf);
      strout[n++]=inf;
    }
// Удаление открывающей скобки
      sp=ReadStackD(sp,inf); continue;
}
```

```
// Если операция
 int pr=priority(ch);
 while (sp != NULL && priority(sp->inf)>=pr)
 {
  sp=ReadStackD(sp,inf);
  strout[n++]=inf;
 }
  sp=AddStack(sp,ch);
}  // end for
```

```
while (sp != NULL)
 {
  sp=ReadStackD(sp,inf);
  strout[n++]=inf;
 }
strout[n++]='\0';
}
```

```
double rasAV(char *str, double *mz)
{
    tstkd *sp=NULL;
    char ch;
    double    inf, inf1, inf2;
```

```
for (unsigned int i=0; i<strlen(str); i++)
{
ch=str[i];
// если операнд
if (ch >= 'A')
{
 sp=AddStack(sp,mz[int(ch)]);
continue;
}
```

```
sp=ReadStackD(sp,inf2); // Если знак операции
sp=ReadStackD(sp,inf1);
 switch (ch)
 {
 case '+': sp=AddStack(sp,inf1 + inf2); break;
 case '-': sp=AddStack(sp,inf1 - inf2); break;
 case '*': sp=AddStack(sp,inf1 * inf2); break;
 case '/': sp=AddStack(sp,inf1 / inf2); break;
 }
}
sp=ReadStackD(sp,inf);
return inf;    }
```

```
int main()
{    double a, b, c, d, f ;
cout << "Vvedite a" << endl;  cin >> masz[int('a')];
cout << "Vvedite b" << endl;  cin >> masz[int('b')];
cout << "Vvedite c" << endl;  cin >> masz[int('c')];
cout << "Vvedite d" << endl;  cin >> masz[int('d')];
cout << "Vvedite f" << endl;  cin >> masz[int('f')];
cout << " Vvedite  virag (a ,b, c, d, f) " << endl;
cin >> str;
AddPostFix(str, strp);
cout << endl << strp;
cout << endl <<" Res = " << rasAV(strp, masz);
  return 0;  }
```