1.Примеры решения задач в структурном подходе

Примеры решения задач с разбиением алгоритма на функции

Примеры применения функций

Напишите функцию f(int N, int& m1, int& m0), которая возвращает первую и последнюю цифры двузначного натурального числа N

```
#include<iostream>
using namespace std;
void f(int N, int& m1, int& m0);
int main()
{int n1, n2, N;
cout<<"Input a two-digit number:"; cin>>N;
f(N,n1,n2);
cout<< "The first digit = "<<n1<<
"\n The second digit = "<<n2<<endl;
system("pause");
return 0;}
```

Код функции

```
void f(int N, int& m1, int& m0)
{
m1=N/10;
m0=N%10;
return;}
```

Примеры применения функций

Напишите функцию f(int N, int * m1, int * m0), которая возвращает первую и последнюю цифры двузначного натурального числа N

```
#include<iostream>
using namespace std;
void f(int N, int* m1, int* m0);
int main()
{int  m1, m2, N;
cout<<"Input a two-digit number:"; cin>>N;
f(N, &m1, &m2);
cout<< "The first digit = "<<m1<<
"\n The second digit = "<<m2<<endl;
system("pause");
return 0;}
```

Код функции

```
void f(int N, int * m1, int* m0)
{
*m1=N/10;
*m0=N%10;
return;}
```

Анализ программы

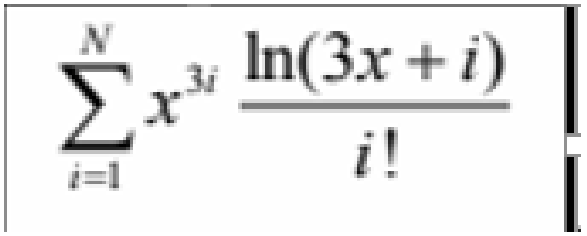
- Если в качестве формальных параметров ***стоят переменные-ссылки***, то для адекватной работы программы необходимо *выделить в вызывающей функции память* под переменные, которые будут переданы по этим ссылкам.
- Вызываемая функция работает непосредственно в той области памяти, которая выделена под переменные в вызывающей функции.

Анализ программы

- Если в качестве формальных параметров стоят ***переменные-указатели***, то для адекватной работы программы необходимо ***разыменовать фактические параметры*** и передать в указатели их адреса, т.е. взять адрес по ссылке
- Вызываемая функция работает непосредственно с разыменованными переменными, поэтому знак указателя необходим при выполнении операций в теле вызываемой функции.

Функция вычисления суммы ряда

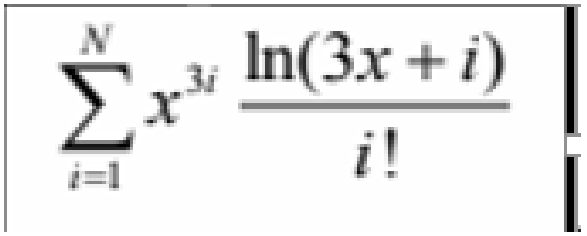
Составить программу для вычисления суммы ряда




```
#include<iostream>
#include<cmath>
using namespace std;
float sum_series(float x, int N);
int main()
{float x,S;
int N;
cout<<"Input x:"; cin>>x;
cout<<"Input N:"; cin>>N;
S=sum_series(x,N);
cout<<"S = "<< S<< endl;
system("pause");
return 0;
}
```

Функция вычисления суммы ряда

Составить программу для вычисления суммы ряда




```
float sum_series(float x, int N)
{float  x3=x*x*x, u=1,s=0;
Int  zn=1;
for (int i=1;i<=N;i++)
     {u*=x3;
     zn*=i;
     s+=u*log(3*x+i)/zn;
     }
return s;
}
```

Функция табулирования суммы ряда для набора аргумента x

Составить программу для построения таблицы значений функции S(x,N), вычисляющей сумму ряда

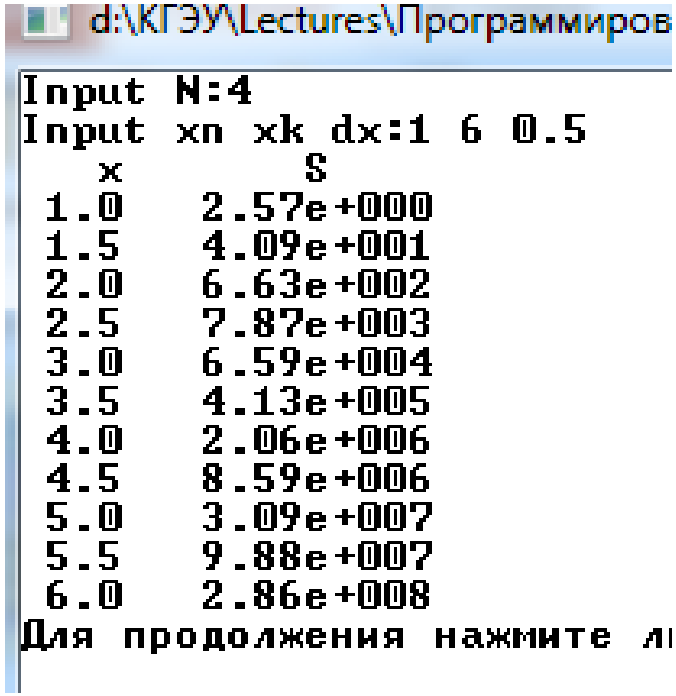
$$\sum_{i=1}^{N} x^{3i} \frac{\ln(3x+i)}{i!}$$

Для $x \in [1;6]$ **с шагом 0,5**

```
#include<iostream>
#include<cmath>
using namespace std;
float sum_series(float x, int N);
void table(float xn, float xk, float dx);
int N;// Gloabal variable
int main()
{float xn, xk, dx;
cout<<"Input N:"; cin>>N; cout<<"Input xn xk dx:";
cin>>xn>>xk>>dx;
table(xn,xk,dx);
system("pause");
return 0;}
```

Код функции table

```
void table(float xn, float xk, float dx)
{
        float S;
        printf("   x       S ");
        for(float x=xn;x<xk+dx/2;x+=dx)
        {
        S=sum_series(x, N);
        printf("\n %3.1f   %.2e", x,S);
        }
        cout<<endl;
        return;
}
```



Передача в функцию массива

Даны два массива. Необходимо найти сумму элементов в каждом из них и вычислить наибольшее среднее арифметическое из средних арифметических обоих массивов.

```
#include<iostream>
#include<cmath>
using namespace std;
void prnt(int c[ ], int n);// вывод массива на консоль
int sum_arr(int c[ ], int n); // сумма элементов массива
float sr_arph(int c[ ], int n); // среднее арифметическое
                                   массива
void test(int c[ ], int n); // проверка способа передачи
массива в функцию
```

Главная программа

```
int main()
{ int a[ ]={1,-2,0,7,-6}, b[ ]={3,7,-2,1,0,4,7};
int sa,sb;
int na=sizeof(a)/sizeof(int),          nb=sizeof(b)/sizeof(int);
prnt (a,na);                           prnt(b,nb);
sa=sum_arr(a,na);                      sb=sum_arr(b,nb);
cout<<"\n sa = "<<sa<<"  sb = "<<sb<<endl;

float sr_ar_max=sr_arph(a,na)>sr_arph(b,nb)?
sr_arph(a,na):sr_arph(b,nb);

cout<<"\nmax_sr_arph = "<<sr_ar_max<<endl;

test(a,na);
prnt(a,na);
system("pause");
return 0;}
```

Коды функций

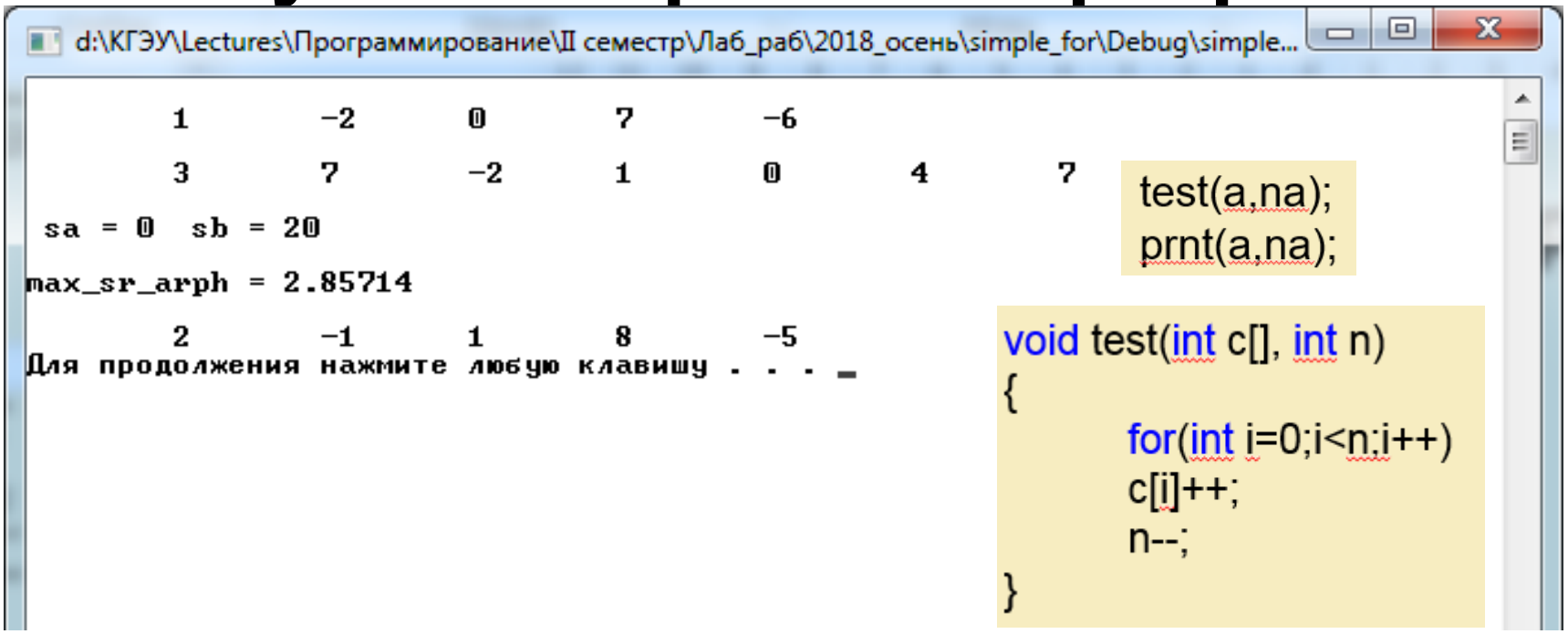
```
void prnt(int c[ ], int n)
{
        cout<<endl;
        for(int i=0;i<n;i++)
                cout<<'\t'<<c[i];
cout<<endl;
return;}

int sum_arr(int c[ ], int n)
{
        int s=0;
        for(int i=0;i<n;i++)
        s+=c[i];
        return s;
}
```

```
float sr_arph(int c[], int n)
{
return sum_arr(c, nc) /float(n);
}

void test(int c[], int n)
{
        for(int i=0;i<n;i++)
        c[i]++;
        n--;
}
```

Результаты работы программы



При использовании массива в качестве аргумента функции происходит ***передача в функцию его адреса***. Это означает, что код внутри функции действует и может изменять настоящее значение массива, используемого при вызове.

Когда массив используется в качестве аргумента функции, *передается только адрес массива, а не копия всего массива*. При вызове функции с именем массива в функцию *передается* ***указатель на первый элемент массива***.

Анализ программы

- ***Передача массива в функцию реализуется по ссылке***, т.е. **копия** массива в функции **не создаётся**, функция работает с тем же участком памяти, где находится массив, описанный в main()
- ***Размер массива – n – передаётся по значению***: для переменных na и nb **создаются копии** в теле функции. Поэтому любые изменения с ними в теле функции не вызывают изменение фактического параметра, значение которого было скопировано в функцию.

Резюме:

- Передача таким же образом двумерных массивов не допускается транслятором.
- Для всех типов массивов допускается передача по указателю на первый элемент массива