Циклические алгоритмы

Цикл - **последовательность операторов, которые выполняются неоднократно.**

Задача. N!

Для заданного значения n вычислить n! (факториал n), то есть найти значение, которое по определению равно

$$n!=1*2*3*\ldots(n-1)*n$$

1!=1

2!=1*2=2

3!=1*2*3=6

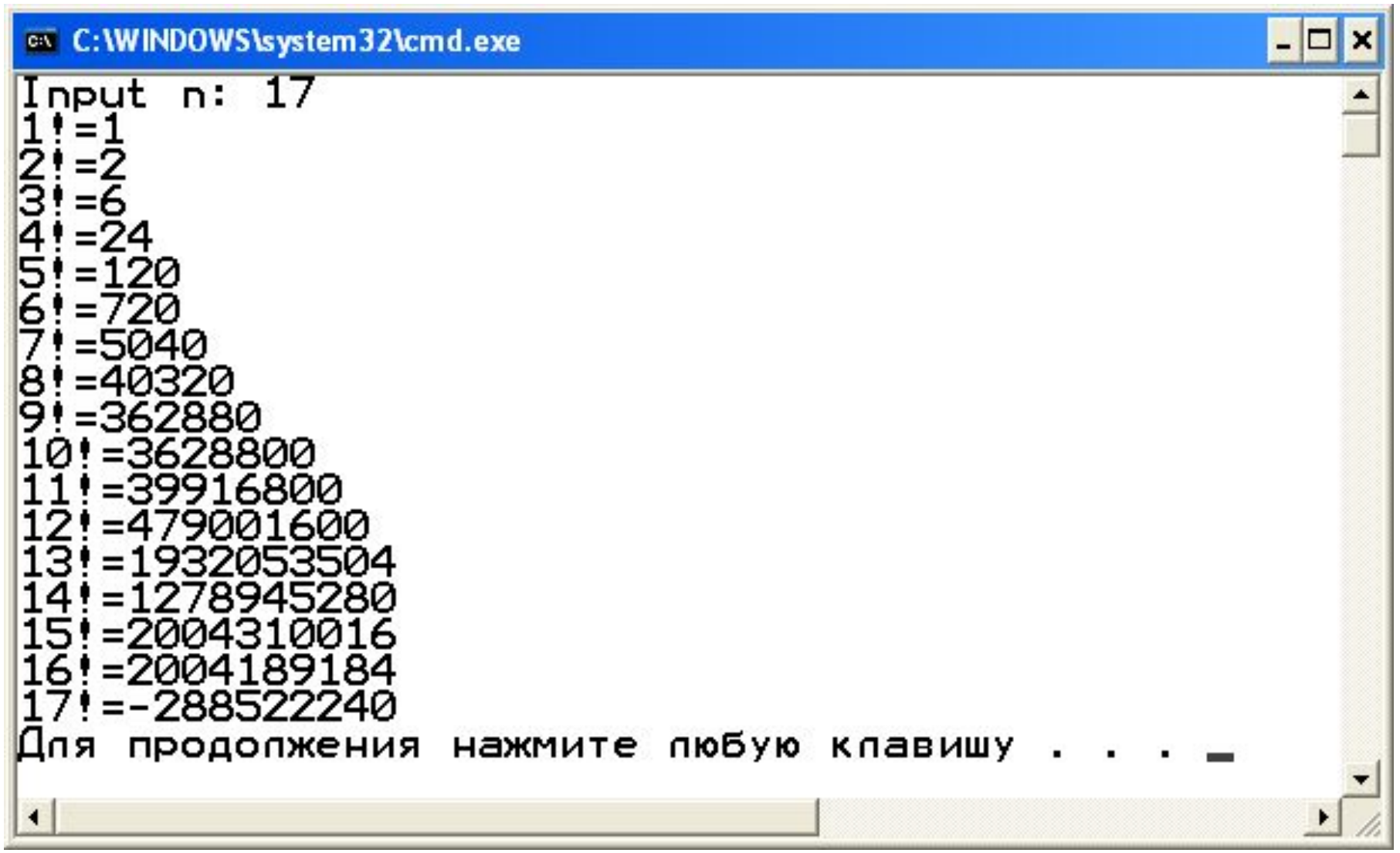
4!=1*2*3*4=24 **4!=3!*4**

n! = (n-1)! * n

Рекуррентная формула – позволяющая вычислить очередное значение числовой последовательности по предыдущему

```
int Factorial = 1;
for(int i=1; i<=n;i++) Factorial=Factorial*i;
```

Задача. N!


C:\WINDOWS\system32\cmd.exe
Input n: 17
1!=1
2!=2
3!=6
4!=24
5!=120
6!=720
7!=5040
8!=40320
9!=362880
10!=3628800
11!=39916800
12!=479001600
13!=1932053504
14!=1278945280
15!=2004310016
16!=2004189184
17!=-288522240
Для продолжения нажмите любую клавишу . . .

Вычисление сумм (ряды Тейлора)

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! \ldots$$

От того, сколько членов степенного ряда сохранено, зависит точность полученного значения.

Задача. **Требуется составить алгоритм для вычисления приближенного значения синуса от произвольного значения аргумента.**

Пусть S_i – значение i-го слагаемого, причем $S_0 = x$. Тогда выполняется следующее соотношение:

$$S_{i+1} = S_i * (-x^2)/(2i*(2i+1))$$

Sum_1

```
double S=x;
double Sum=x;
for(int i=1;i<=5;i++)
{
   S=-S*pow(x,2)/(2*i*(2*i+1));
   Sum=Sum+S;
}
```

Почему 5 ?

Погрешность вычислений – допустимое отклонение вычисляемого значения

$$| S_{i+1} - S_i | < eps$$

Sum_2

```
double const eps=0.0001; //погрешность
int i=0;
double S0=x; //текущий член ряда
double S1=x; //следующий член ряда
Sum=x;
do
 {
   i++;
   S0=S1;
   S1=-S0*pow(x,2)/(2*i*(2*i+1));
   Sum=Sum+S1;
 }
while (fabs(S0-S1)>eps);
```

Алгоритмы обработки массивов

- подсчет количества элементов, обладающих заданным свойством
- поиск максимального и минимального элементов
- поиск элементов, обладающих заданным свойством
- поиск подпоследовательностей
- сортировка элементов

Алгоритмы обработки массивов

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
//1. количество четных положительных
int count=0;
for (int i=0;i<n;i++)
   if (a[i]>0 && a[i]%2==0) count++;
```

```
//2. сумма всех отрицательных
int sum=0;
for (int i=0;i<n;i++)
   if (a[i]<0) sum=sum+a[i]; //или sum+=a[i]
```

Алгоритмы обработки массивов

```
//3. последовательный поиск
//найти индекс элемента со значением key
int key=5; //искомое значение
int index=-1; //если элемента key в массиве нет
for(int i=0;i<n;i++)
{
  if (a[i]==key) index=i;
  break; //прервать выполнение цикла
}
```

6	13	14	25	33	43	51	53	64	72	84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Алгоритмы обработки массивов

```
//4. найти первый отрицательный элемент в массиве
int index=-1; //если в массиве нет отрицательных
int i=0;
while (i<n && a[i]<=0) i++;
if (i<n) index=i;
```

6	13	14	-25	33	43	51	-53	64	72	-84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
//5. поиск максимального элемента массива
int max=a[0];
for(int i=1;i<n;i++)
if (a[i]>max) max=a[i];
```

Алгоритмы обработки массивов

```
//4. найти первый отрицательный элемент в массиве
int index=-1; //если в массиве нет отрицательных
int i=0;
while (i<n && a[i]<=0) i++;
if (i<n) index=i;
```

6	13	14	-25	33	43	51	-53	64	72	-84	93	95	96	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
//5. поиск максимального элемента массива
int max=a[0];
for(int i=1;i<n;i++)
if (a[i]>max) max=a[i];
```

Алгоритмы обработки массивов

1	3	0	0	0	43	0	0	64	72	-84	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
//6. поиск подпоследовательности
//самая длинная цепочка нулей
int length=0;
int maxlength=length;
int i=0;
while (i<n)
{
  while(a[i]==0 && i<n) { length++; i++;}
  if (length>maxlength) maxlength=length;
  length=0;
  i++;
}
```