

## Шаблоны функций

```
Template< class T>  
Class pair  
{  
    T a,b;  
}
```

- *Шаблон функции представляет собой обобщенное определение функции, из которого компилятор автоматически создает представитель функции для заданного пользователем типа (или типов) данных.*

*Синтаксис объявления шаблона функции имеет следующий вид:*

**Template <class T1|T1 идент1, class T2|T2 идент2, class Tn|Tn идент n>**

**Возвр\_тип имя\_функции (список параметров)**

**{**

**//Тело функции**

**}**

```
#include <iostream>  
using namespace std;  
template <class T>  
T Sqr(x)  
{  
    return x*x;  
}
```

```
template <class T>  
T* Swap(T* t, int ind1, int ind2)  
{  
T tmp=t[ind1];      // t строчные  
T[ind1]= t[ind2];  
T[ind2]=t[ind1];  
Return t;  
}
```

```
template <typename T1, typename T2>  
void Display( T1 x, T2 y)  
{  
    cout.width(8);  
    cout<<x<<' ';<br>  
    cout.width(8);  
    cout<<y<<endl;<br>  
}
```

```
template<class T, int offset>  
void GetAddres(T* obj, unsigned int  
    *pAddr)  
{  
    *pAddr=(unsigned  
        int)&obj[0]+offset*sizeof(T);  
}
```

```
int main()  
{  
int n=10,sq_n, i=2, j=5;  
double d=10.21, sq_d;  
char* str=" Шаблон";  
sq_n=Sqr(n);  
sq_d=Sqr (d);  
int Arr[100];  
unsigned int addr=0;
```



```
cout<<"значение n="<<n<<endl<<"его  
    квадрат="<<sq_n<<endl;  
cout<<"Значение d="<<d<<endl  
<<"Его квадрат="<<sq_d<<endl;  
cout<<"Исходная строка=" " " <<str<<" "  
    "<<endl;  
cout<<"Преобразованная строка =" "  
    <<Swap (str,l,j) << " " " " <<endl;
```

```
Display(n,d);  
Display(sq_n,sq_d);  
GetAddres< int , 5>(Arr,&addr);  
Cout<<"Адрес элемента  
    Arr[5]="<<hex<<showbase<<uppercase<<ad  
    dr;  
Return 0;  
}
```

*Как и для обычных функций, можно создать прототип шаблона функции в виде его предварительного объявления. Например:*

```
Template<class T>  
T* Swap (T* t, int ind1, int ind2);
```

*Имена параметров шаблонной функции в ее объявлении могут не совпадать.*

## **ПЕРЕГРУЗКА ШАБЛОНОВ ФУНКЦИЙ**

```
#include <iostream.h>  
using namespace std;  
//возвращает больший из двух  
параметров  
template<class t>  
const T& max(const T& a, const T& b)  
{  
return a>b?a:b;  
}
```

*//возвращает наибольший элемент массива*

```
template<class T>
```

```
const T max( T* a, size_t size)
```

```
{
```

```
T* tmp=a;
```

```
For(int i=0;i<size;i++)
```

```
{
```

```
if (a[i]>*tmp)
```

```
*tmp=a[i];
```

```
}
```

```
return *tmp;
```

```
}
```

```
int main()  
{  
int m=9, n=12;  
int arr[]={3,5,7,9};  
cout<<"max int="<<max(m,n)<<endl;  
cout<<" max in  
    arr="<<max(arr,sizeof(arr)/sizeof(int))  
<<endl;  
return 0;  
}
```

при выполнении программа выводит на  
экран

**max int=12**

**max in arr=9**