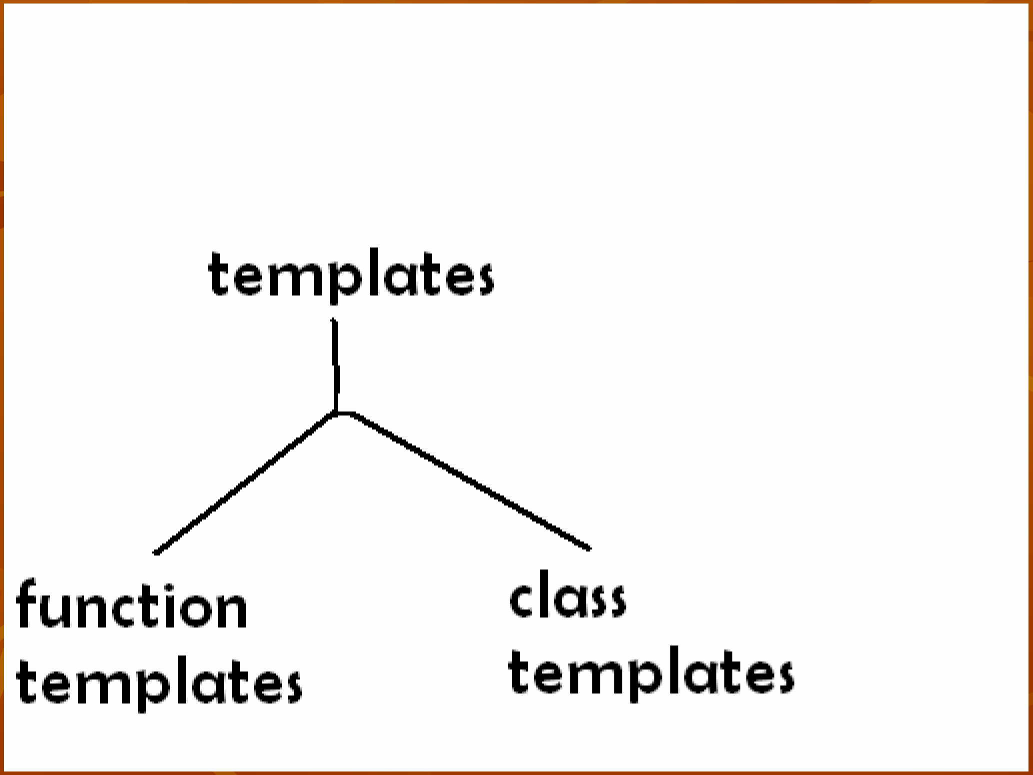
Templates

Template is one of the features of c++ which enables us to define generic classes and functions and thus provides support for generic programming.

A template can be considered as a kind of macro. When an object of specific type is defined for actual use, the template definition for that class is substituted with the required data type .


templates
function templates
class templates

Function Templates

- Functions operate on a particular data type. It can be overcome by defining that function as a function template or generic function.

```
#include <iostream.h>
void swap(char & x, char & y)
{
  char t;
  t=x;
  x=y;
  y=t;
}

void swap(int & x, int & y)
{
  int t;
  t=x;
  x=y;
  y=t;
}
```

```
Void swap(float & x,float & y)
{
  float t;
  t=x;
  x=y;
  y=t;
}
Void main()
{
  char ch1,ch2;
  cout << “Enter two characters<ch1 ,ch2>:”;
  cin >> ch1 >> ch2;
  swap (ch1,ch2);
  cout<< “On swapping < ch1 ,ch2>:” << ch1 << “  ” << ch2<<endl;
  int a ,b;
  cout << “Enter two integers<a,b>:”;
  cin >> a >>b;
  swap(a,b);
  cout<< “On swapping <a ,b>:”<<a << “ ” <<b<<endl;
  float c,d;
  cout<< “Enter two floats<c,d>:”;
  cin>>c>>d;
  swap(c,d);
  cout<<“On swapping < c,d>:”<<c<<“ ”<<d<,endl;
}
```

Output

```
Enter two characters<ch1,ch2> :          R K
On swapping < ch1,ch2>:                  K R
Enter two integers<a ,b>:                5 10
On swapping <a,b>:                       10 5
Enter two floats< c,d >:                 20.5 99.5
On swapping <c,d>:                       99.5 20.5
```

```
#include <iostream.h>

template<class T>
Void swap(T & x,T & y)
{
  T t;
  t=x;
  x=y;
  y=t;
}
```

such functions are known as function templates.

Another function template for finding maximum of two data items:

```
template <classT>
T max(T a,T b)
{
  if(a>b)
     return a;
  else
     return b;
}
```

The function template is invoked in the same manner as a normal function :

```
x = max(y, z);
```

Function Template with Multiple Parameters

Syntax

template < class T1 ,class T2,- - - ->

Return type function name(arguments of types T1,T2,- - -)

{

------------ (Body of function)

}

```
#include<iostream.h>
#include<string.h>
template<class T1,class T2>
void display(T1 x,T2 y)
{
   cout<<x<<“ ”<<y<<endl;
}
int main()
{
   display(1999, “EBG”);
   display(12.34,1234);
```

Class Templates

```
template < class T>
class classname
{
    ------------------// class member specification
    ----------------//with anonymous type T
    ---------------//wherever appropriate
}
```

A class created from class template is called template class. The syntax for defining an object of a template class is:

```
classname <type> objectname(arglist);
```

```
#include<iostream.h>
const size=3;
template<class T>
class vector
{
 T * v;
Public: vector()
{ v= new T[size];
for(int i=0; i<size; i++)
   v[i]=0;
}
vector(T *a)
{
  for(int i=0; i<size; i++)
     v[i]=a[i];
}
```

```
T operator * (vector & y)
{ T sum =0;
for(int i=0; i<size; i++)
sum+=this->v[i]*y.v[i];
return sum;
}
};

int main()
{
int x[3]={1,2,3};
int y[3]={4,5,6};
vector <int> v1;
vector <int> v2;
v1=x;
v2=y;
int R= v1* v2;
```

```
cout<<"R= " <<R<<endl;
return 0;
}
```

Ouptut:

```
R =32
```

Class templates with multiple parameters

```
Template classname
{
   -------------------------
  ---------------------------(Body of the class)
}
```

```
#include<iostream.h>

template<class T1,class T2>
class Test
{
T1 a;
T2 b;

public:
Test(T1 x,T2y)
{
a=x;
b=y;
}
void show()
{
    cout<<a << " and" <<b<<endl;
}
};

int main()
```

```
{
Test< float, int> test1(1.23,123);
Test <int, char> test2(100,'W');
test1.show();
test2.show();
return 0;
}
```

Output

```
1.23 and 123
100 and W
```