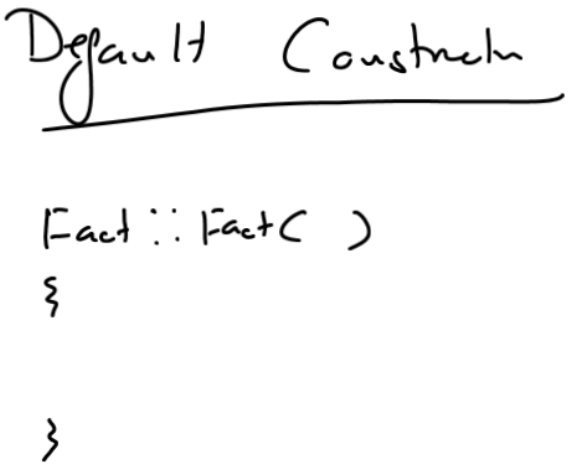
```
#include <iostream.h>
#include <conio.h>
class Fact
{
   int n;
   int f;
public:
          Fact();
          void get();
          void calculate();
          void show();
};
Fact::Fact()
{
    f=1;
}
void Fact::get()
{
  cout<<"Enter an iont:";
  cin>>n;
}
void Fact::calculate()
{
  for(int i=1;i<=n;i++)
          f=f*i;
}

void Fact::show()
{
  cout<<"No is "<<n<<endl;
 cout<<"Its fact is "<<f;
}
int main()
{
   clrscr();
   Fact obj;
   obj.get();
   obj.calculate();
   obj.show();
   getch();
   return 0;
}
```

```
#include <iostream.h>
#include <conio.h>
class Fact
{
   int n;
   int f;
public:
          void init();
          void get();
          void calculate();
          void show();
};
void Fact::init()
{
    f=1;
}
void Fact::get()
{
  cout<<"Enter an iont:";
  cin>>n;
}
void Fact::calculate()
{
  for(int i=1;i<=n;i++)
          f=f*i;
}

void Fact::show()
{
  cout<<"No is "<<n<<endl;
 cout<<"Its fact is "<<f;
}
int main()
{
   clrscr();
   Fact obj;
   obj.init();
   obj.get();
   obj.calculate();
   obj.show();
   getch();
   return 0;
}
```



Default Constructor
==========

1. In C++ there is a special rule regarding constructors and the rule is that if a programmer DOES NOT DEFINE any CONSTRUCTOR in his class then the C++ compiler automatically inserts a special constructor in our class called as **DEFAULT CONSTRUCTOR** .

2. The default constructor generated by the compiler has a blank body. For example: If the name of the class is **Fact** then the compiler generated default constructor will be

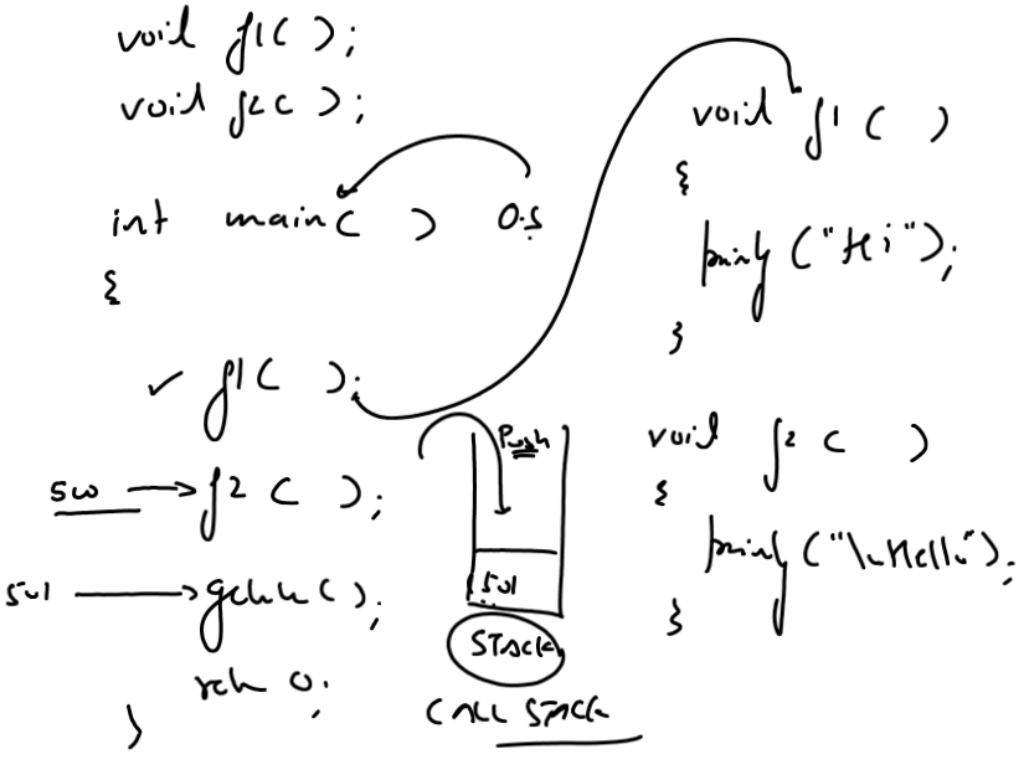
```
Fact::Fact()
{
}
```

3. As we can observe the default constructor has an empty body so we also call it DO NOTHING CONSTRUCTOR.

4. We must remember that the default constructor is provided by the compiler only when we haven't created any constructor ourselves in the class.

5. It means in a single class we can't have default constructor as well as programmer define constructor together.

6. Since the default constructor doesn't perform any useful tast for the programmer so it is recommended that we should replace the default constructor with our own constror and perform some useful task in it.

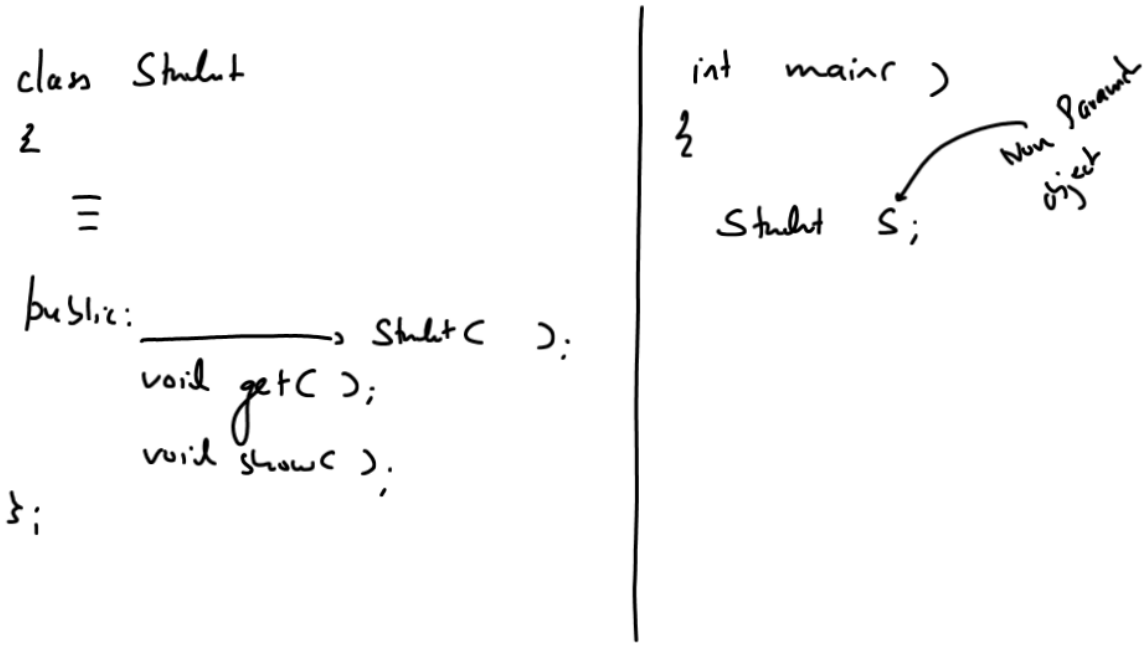


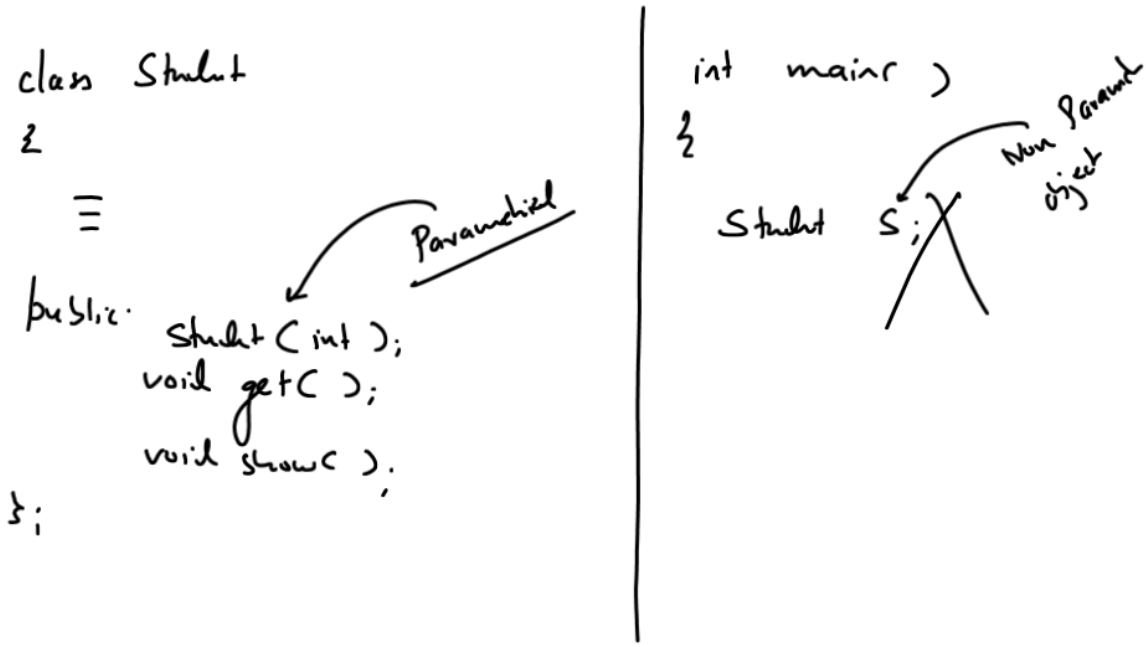
Creating Parametrized Constructor
==========
1. Just like we can have parametrized member functions similarly we also can have PARAMETRIZED CONSTRUCTOR.

2. In other words it means that **CONSTRUCTORS CAN ACCEPT ARGUMENTS**

3. But if we have a parametrized constructor in the class then there is a very impt rule we must follow

4. The rule is that if our class contains only a parametrized constructor then each and every object of the class which we will create must also be parametrized.






```
#include <iostream.h>
#include <conio.h>
#include <string.h>
class Emp
{
   int age;
   char name[20];
   float sal;
public:
      Emp(int,char*,float);
      void show();
};
Emp::Emp(int a,char *p,float s)
{
    age=a;
    strcpy(name,p);
    sal=s;
}
```

F
a 24
n "Rahul"
s 30000.0

F
a 25
n "Amit"
s 28000.0

```
void Emp::show()
{
cout<<age<<","<<name<<","
<<sal<<endl;

}

int main()
{
    clrscr();

✓ Emp E(24,"Rahul",30000.0);
✓ Emp F(25,"Amit",28000.0);
✓ E.show();
    F.show();
    getch();
    return 0;
}
```

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
class Emp
{
   int age;
   char name[20];
   float sal;
public:
      Emp(int,char*,float);
      void get();
      void show();
};
 void Emp::get()
 {
     cout<<"Enter age,name and sal:";
    cin>>age>>name>>sal;
}
Emp::Emp(int a,char *p,float s)
{
    age=a;
    strcpy(name,p);
    sal=s;
```

```
void Emp::show()
{
cout<<age<<","<<name<<","
<<sal<<endl;

}

int main()
{
    clrscr();
✓   Emp E(24,"Rahul",30000.0);
✗   Emp F;
    F.get();
    E.show();
    F.show();
    getch();
    return 0;
}
```