```
#include <bits stdc++.h>
using namespace std;

#define pi 3.14

int main()
{ int r = 10;
  int area = pi*r*r;
  Cout << area;
}
```

```
#include <iostream.h>
using namespace std;
typedef pair<int,int> pr
int main()
{
  ~~pr p~~
  ~~pr p. first = 10;~~

}
```

Tertiary operators

```
int c =  (a>b) ?  a  :  b;
```

if this condn is true then 'a' is assigned to c

b is assigned to c if condn fails.

~~[illegible]~~ Inline functions

```
#include <iostream.h>
using namespace std;
inline int max (int a, int b)
{
  return (a>b)? a:b;
}
int main(){
 int a, b;
 cin>>a>>b;
 int c = max (a,b);
 int x, y;
 x= 12;
 y=34;
 int z= max (x,y);
}
```

The content inside the function is copied where the f^n is called in the main prog. All the parameters are taken care of.

Advantages - Code becomes a bit fast.

Disadvantages - The output file becomes bulky. and large.

Why every fun^n can't be inline?

As compiler only allows those f^n to be inline which are called 2-3 times in the prog.

Default arguments

```
#include <iostream>
using namespace std;
int sum(int a[], int size, int si = 0)
{
  int ans = 0;
  for(int i = si; i< size; i++)
  {
    ans += a[i];
  }
  return ans;
}
int main(){
int a[20];
Cout << sum(a, 20).<<endl;
}
```

This is called default arguement. This parameter is handled if user does not gives the 'last' parameter.

here user has not passed the 'si' hence default arguement places the value of si as 0

Note

Default arguments can only be placed from right hand side and not anywhere from middle.

int sum(int a[], int size = 10) int si) ✗

int sum (int a[], int size = 10, int si = 10), int si) ✗