C++ in depth

Functions in c++

Saurabh Shukla (MySirG)

Agenda

1. inline
2. default argument
3. reference variable
4. call by value, call by address, call by reference

inline

Function in a program to save memory space which becomes appreciable when a function is likely to be called many times.

However, everytime a function is called, it takes lot of extra time in executing a series of instructions for tasks such as jumping to the functions, saving registers, pushing arguments into the stack and returning to the calling function.

So, when function is small it is worthless to spend so much extra time in such tasks in cost of saving comparatively small space.

To eliminate the cost of call to small functions, C++ proposes a new feature called inline function.

An inline function is a function that is expanded in a line when it is invoked.

Compiler replaces the function call with the corresponding function code

inline is a request not a command

So the compiler may ignore the request in some situations:

Few of them are:

- Function containing loops, switch or goto
- Function with recursion
- containing static variable

```
inline int add(int, int);


int add(int x, int y)
{
    return x+y;
}
```

default arguments

- We can set default values to the arguments in a function to let them allow to invoke it without passing value to the corresponding receiving variable.

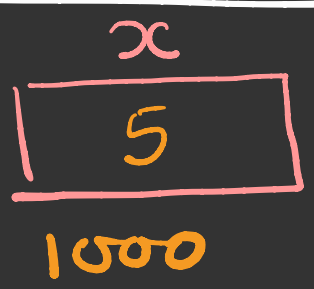
- It is not necessary that all arguments should have some default values.
- During declaration of function, you can set default values to the arguments.
- Function cannot have non default argument after a default argument.

Formal Arguments

1. Ordinary variables — C/C++
2. Pointer variables — C/C++
3. Reference variable — C++ (V.Imp)

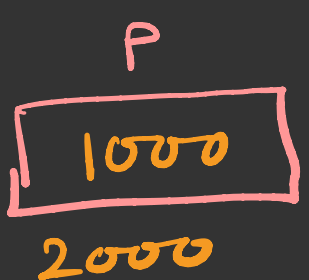
Reference Variable

```
int x = 5;
```



ordinary variable

```
int *p;
p = &x;
```



Pointer variable

```
int &y = x;
```

Reference variable

जल अक्षत
पानी चावल

Call by value | Call by address | Call by reference

```
void f1(int,int);
int a=3,b=4;
f1(a,b);
```

a b
3 4

↖ call by value

```
void f1(int x, int y)
{
}
```

x y
3 4

```
}
```

```
void f2(int*,int*);
int a=3,b=4;
f2(&a,&b);
```

a b
3 4
1000 2000

↑ ~~Call by reference~~
Call by address

```
void f2(int *p, int *q)
{
```

p q
1000 2000

```
}
```

```
void f3(int&, int&);
int a=3,b=4;
f3(a,b);
```

a b
3 4
1000 2000

↑ Call by reference

```
void f3(int &m, int&n)
{
```

m n
1000 2000

```
}
```

```
scanf("%d", &x);

cin>>x;
```