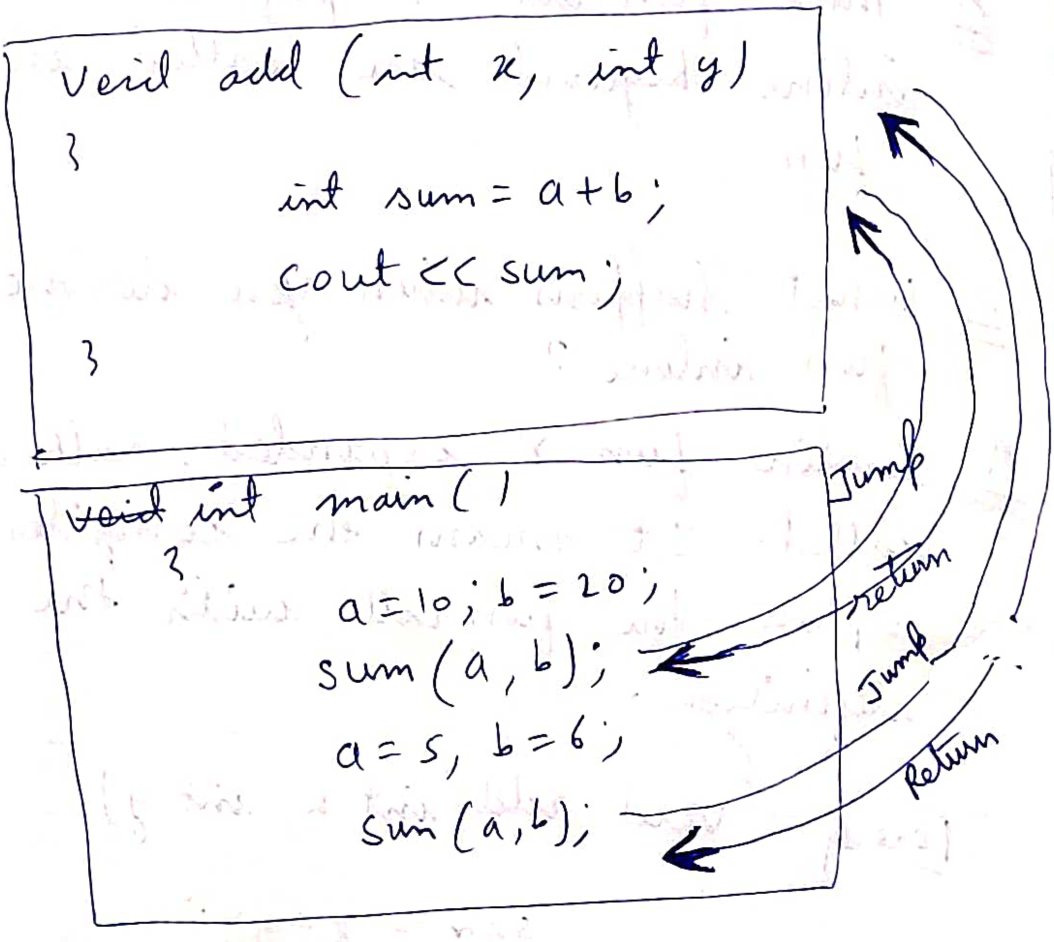
Inline Functions

Q How does a normal function call works

A During a normal fun call, the program will jump to the fun & then return back.

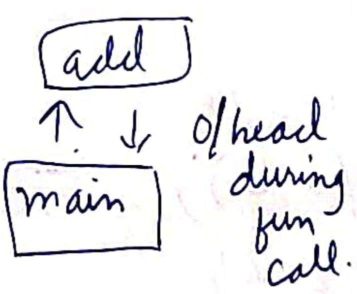
eg

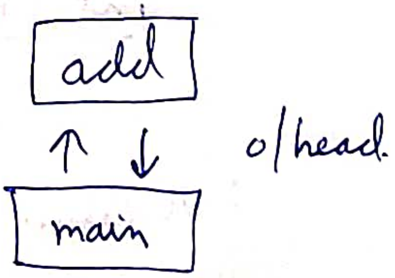


```
void add (int x, int y)
{
        int sum = a+b;
        cout << sum;
}

~~void~~ int main ()
  {
        a=10; b=20;
        sum(a,b);
        a=5, b=6;
        sum(a,b);
```

- This jump & return causes an overhead. Means time is wasted during jump.





Q How can you save o/head during fun call?

Ans By using inline fun.

Q What are inline functions?

A Those fun wh. r prefixed with the inline keyword are called inline fun.

Q What happens when you declare a fun inline?

A Inline fun r expanded, rather than called. It means the compiler will replace the fun call with the fun definition.

Fore eg

```
void add(int x, int y)
{
    sum = x+y;
    cout << sum;
}
```

int main()

{

a=10, b=20;

Code copied ~~sum(a,b);~~ → sum=x+y cout<<sum }

a=5, b=6;

Code copied ~~sum(a,b);~~ → sum=x+y cout<<sum;

}

Q Advantage of inline?

A There is no ~~fun~~ jump wh. occurs during fun call. So there is no overhead. ∴ inline fun r faster.

Q ~~Significant~~ Are inline fun significant?

A No. Inline is a request to compiler not a command.

- Compiler will only inline short fun & ignore inline keyword for large fun codes.

Q Disadvantage of ~~comp~~ inline?

A Your code size increases as the code is copied several times.

Q WAP to inline a fun without class?

Ans ~~Inline~~

```
#include...
 Using...
inline void add(int x, int y)
{    int sum = x+y;
         cout << sum;

}
int main( )
{       int a=10, b=20;
        sum(a, b);

}
```

Q WAP to inline all fun inside a class?

A // use inline Keyword.

```
#...
using...
class student
{
        private:
         int rn, m1, m2;
        public:
inline void set(int x, int y, int z);
        inline void get();
        inline void add();
};

inline void student:: set(it x, it y, it z){
{
          rn = x,
          m1=y;
          m2=z;

}
inline void student:: get ()
{
          cout << rn << m1 << m2;
}
inline void add ()
{
          cout << m1+m2;
}
```

```
int main ()
{
        student obj1;
        obj1.set (1, 10, 20); ——> rn=x
                                   m1=y
                                   m2=z
        obj1.get (); ——> ≡
        obj1.add (); ——> ≡
}
```

Q WAP to inline all fun inside a class w/o using inline keyword?

A • When you define a fun inside class it will be automatically inlined. No need to use inline keyword.

```
#...
using...
class student
{
        private:
         int rn, m1, m2;
        public:
        void set (int x, int y, int z)
        {
                rn = x;
                m1=y;
                m2=z;
        }
        void get ()
        {
                cout << rn << m1 << m2;
        }
        void add ()
        {
                cout << m1+m2;
        }
};
```

```
int main( )
?
        student obj1;
         obj1. ~~ad~~set (5, 10, 20);
          obj1.get( );
          obj1.add( );

}
```

This pointer

Q What is this pointer?

A
- Every obj has a ptr to itself. It is called this ptr.
- When any obj calls a fun, that obj is called calling obj.
- [illegible] ~~can~~ Using this ptr, variables of calling obj can be accessed.

Q What is the use of this ptr.?

A
1) this ptr is used in operator o/loading to return the calling obj.
2) this ptr is used to resolve ambiguity when the fun parameters have same name as class variables

Q WAP to show the use of this ptr?

A (Next page)

```
#
in
Class student
{
        private:
        int x, y, z;   // These variable
                            are hidden
        public:
        void set (int x, int y, int z);
         void get ();

};
Void student :: set (int x, int y, int z)
{
            this->x = x;
These are   this->y = y;      These r fun parameters
class var   this z = z;

 }
void  student :: get ( )
 {
          Cout << x << y << z;

  }
 Void  student :: add ()
  {
           cout << x+y+z;

   }

int main ()
{
        student obj1;
          obj1.set (1, 10, 20);
          obj1.get( )

  }
```

```
 1 , 10, 20
 ↓    ↓   ↓
 x    y   z
 ↓    ↓   ↓
this->x  this->y  this->z
```

Static Keyword

Q Where is static keyword used?

A Static kw can be used w/ local, global or class variables.

Q What happens when you declare a class variable as static?

A - All objects will share 1 copy of that variable.
- Any change made by any obj will affect the value for all objects.

Q WAP to show the use of static kw & in a class / static variables?

A
- Static variable can be used to keep count of no. of objects created and destroyed.
- Use a static int count variable
- Use constructor to increment value of count when obj is created.
- Use destructor to decrement " " " " obj is destroyed.

```
class student
{   private:
    static int count; //use Static Prefix
    int ~~ron~~ rn, m1, m2;
    public:
    void set (int x, int y, int z);
     student ();
    ~student ();

};
```

```
int student :: count ***
```

//static variable is only declared ~~XXX~~ inside a class. It is not defined. So here ~~static~~ count variable is defined outside the class.

```
void student :: set (int x, int y, int z)
{
        rn = x;
         m1 = y;
          m2 = z;

 }
student :: student ()
 {
        count ++; // ~~destructor~~ Constructor will increment
                  the value of count;

        Cout << "Const called"
             << "obj created"
              << " count = " << count;


}
```

```
student :: ~student ( )
{
        count - - ;
        Cout << " dest called " <<
                 " obj destroyed <<
                 " count of obj = " << count ;
}

int main ( )
{
        student  obj1 , obj2 , obj3 , obj4 ;

}
```

```
O/P :-  Const called . Obj created . Count = 1
           "       "      "       "       "    = 2
           "       "      "       "       "    = 3
            "     - -    - -    -   -     -    = 4
          - -  - -  - -  - -  - -  -           = 5

        Dest  called  " destroyed  cnt = 4
          .      .         '       '   = 3
          '      '         '       '   = 2
          '      '         '       '   = 1
          '      '         '       '   = 0.
```