OOPS

Inline Functions

Definition

An inline function is a function that is expanded in line when it is invoked. Inline expansion makes a program run faster because the overhead of a function call and return is eliminated. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. ***This substitution is performed by the C++ compiler at compile time***. Inline function may increase efficiency if it is small. It is defined by using keyword "***inline***"

Why Inline Functions?

- One of the objectives of using functions in a program is to save some memory space, which becomes appreciable when a function is likely to be called many times.
- Every time a function is called, it takes a lot of extra time in executing a series of instructions for tasks such as jumping to the function, saving registers, pushing arguments into the stack, and returning to the calling function.
- When a function is small, a substantial percentage of execution time may be spent in such overheads.
- One solution to this problem is to use macro definitions, known as macros. Preprocessor macros are popular in C.
- The major drawback with macros is that they are not really functions and therefore, the usual error checking does not occur during compilation.
- C++ has different solution to this problem. To eliminate the cost of calls to small functions, C++ proposes a new feature called inline function.

Syntax

```
inline return-type function-name(parameters)
{
    // function code
}
```

Properties

- Inlining is only a request to the compiler, not a command.
- **Compiler can ignore the request for inlining**.
- Compiler may not perform inlining in such circumstances like:
    - If a function contains a **loop**. (for, while, do-while)
    - If a function is **recursive**.
    - If a function contains **static** variables.
    - If a function return type is other than **void**, and the return statement doesn’t exist in function body.
    - If a function contains **switch** or **goto** statement.

Example 1

```
#include <iostream>
using namespace std;
inline int square(int a) { return a * a; }
int main() {
   cout << "The Square of 5 is " << square(5) << endl;
   return 0;
}
```

Output

```
The Square of 5 is 25
```

Example 2 (Inline Functions with Classes)

- It is also possible to define the inline function inside the class.
- **All the functions defined inside the class are implicitly inline.**
- If you need to explicitly declare inline function in the class then just declare the function inside the class and define it outside the class using inline keyword.

```
// Demonstrating Inline Functions
#include <iostream>
using namespace std;
class azcalc {
   int a, b, add, sub;
  public:
   void getInput();
   void addition();
   void subtract();
};
inline void azcalc ::getInput() {
   cout << "Enter first value : ";
   cin >> a;
   cout << "Enter second value : ";
   cin >> b;
}
inline void azcalc ::addition() {
   add = a + b;
   cout << "Addition of two numbers : " << a + b << "\n";
}
inline void azcalc ::subtract() {
   sub = a - b;
```

```
   cout << "Difference of two numbers : " << a - b << "\n";
}
int main() {
   azcalc s;
   s.getInput();
   s.addition();
   s.subtract();
   return 0;
}
```


NO NOTES TO SHOW