Computer Programming (a)

E1123

Fall 2022-2023

Lecture 8

Functions – part 2

INSTRUCTOR

DR / AYMAN SOLIMAN

➢ Contents

➢Function Declaration and Definition

- A C++ function consist of two parts:

Declaration: the return type, the name of the function, and parameters (if any)

Definition: the body of the function (code to be executed)

- void **myFunction()** {   // **declaration**

  the body of the function   //(**definition**)

  }

➢Parameters and Arguments

```
void myFunction(string fname) {
  cout << fname << " Ayman\n";
}

int main() {
    myFunction(”Mahmoud");
    myFunction("Jana");
    myFunction(”Zyad");
    return 0;
}

// Mahmoud Ayman
// Jana Ayman
// Zyad Ayman
```

- When a **parameter** is passed to the function, it is called an **argument**. So, from the example above: fname is a **parameter**,
- while Mahmoud, Jana and Zyad are arguments.

➢C++ Default Parameters

➢ You can also use a default parameter value, by using the equals sign (=).

If we call the function without an argument, it uses the default value ("Norway"):

```
void myFunction(string country = "Norway") {
  cout << country << "\n";
}

int main() {
  myFunction("Sweden");
  myFunction("India");
  myFunction();
  myFunction("USA");
  return 0;
}
```

```
// Sweden
// India
// Norway
// USA
```

➢C++ Multiple Parameters

➢ Inside the function, you can add as many parameters as you want:

```
void myFunction(string fname, int age)
{
  cout << fname << " Ayman. " << age << " years old. \n";
}

int main() {
  myFunction(”Mahmoud", 10);
  myFunction("Jana", 8);
  myFunction(”Zyad", 4);
  return 0;
}

// Mahmoud Ayman. 10 years old.
// Jana Ayman. 8 years old.
// Zyad Ayman. 4 years old.
```

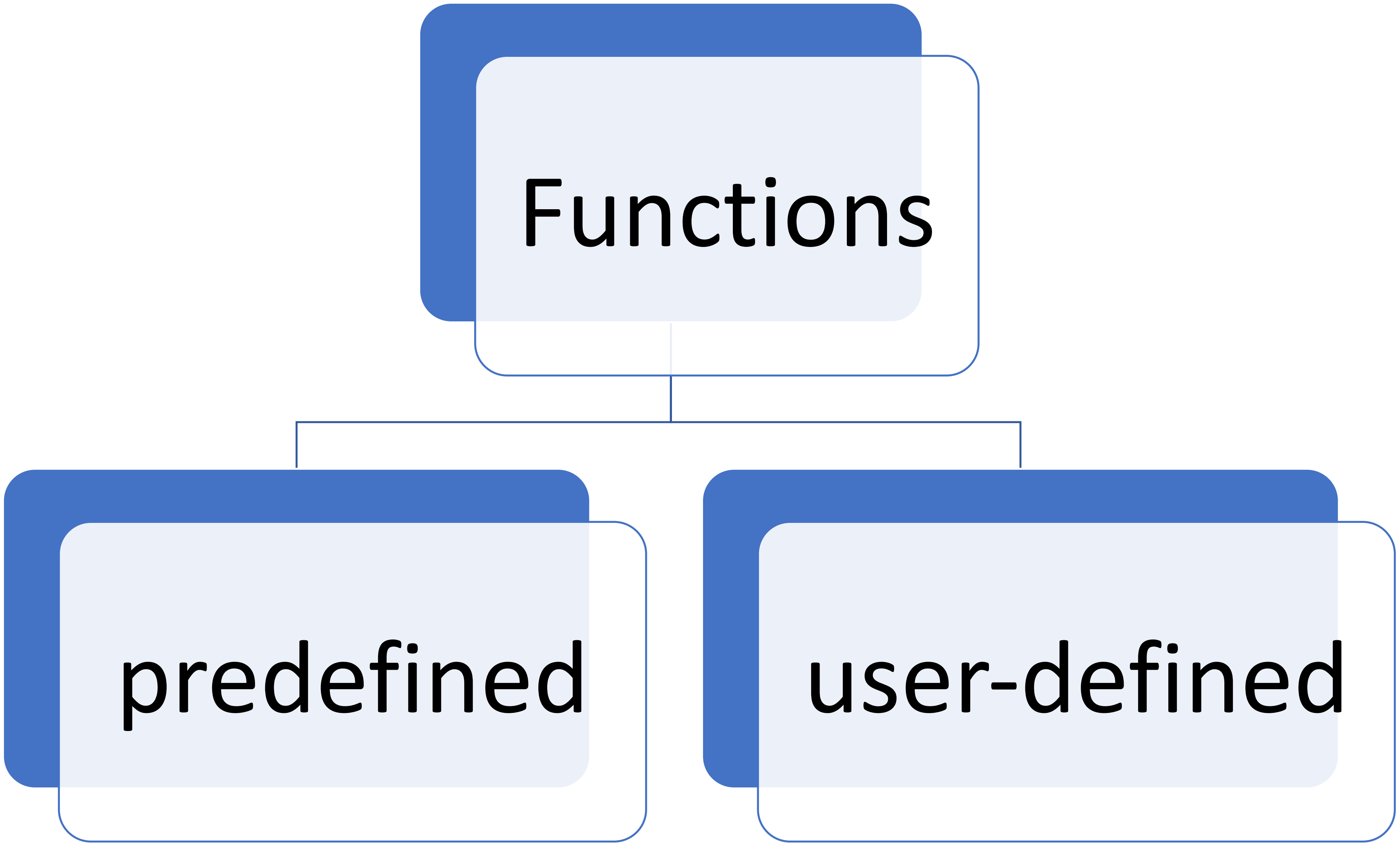
Note that when you are working with multiple parameters, the function call must have the same number of arguments as there are parameters, and the arguments must be passed in the same order.

➢C++ Recursion

➢ Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

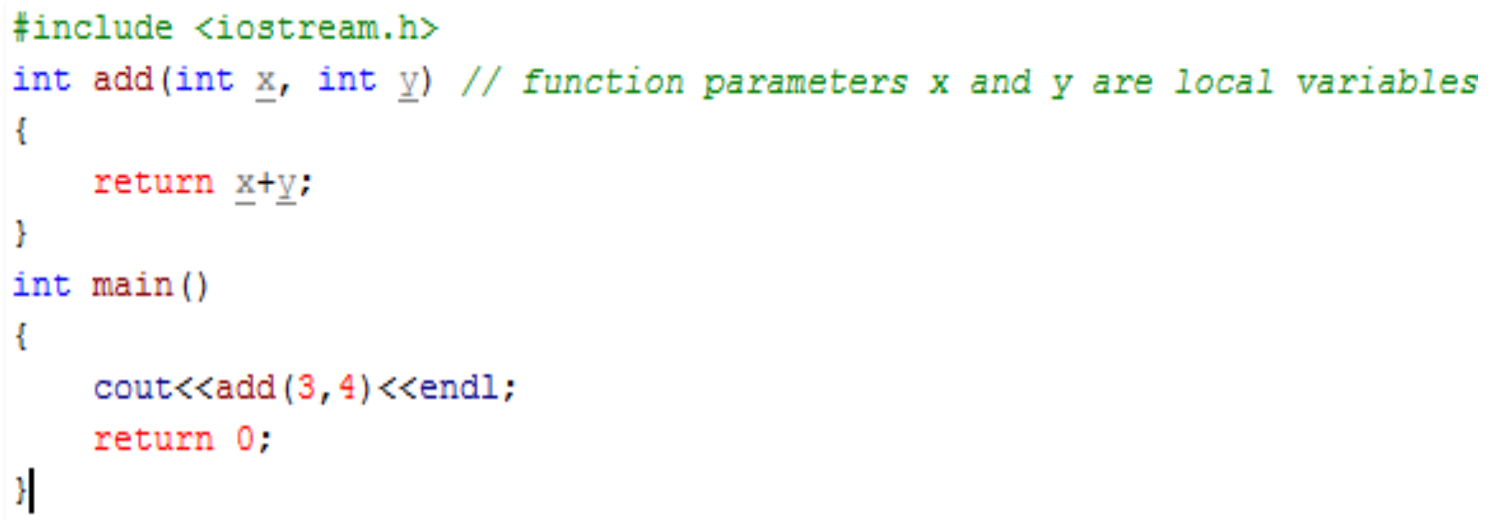
```
int sum(int k) {
  if (k > 0) {
    return k + sum(k - 1);
  } else {
    return 0;
  }
}

int main() {
  int result = sum(10);
  cout << result;
  return 0;
}
```

```
10 + sum(9)
10 + ( 9 + sum(8) )
10 + ( 9 + ( 8 + sum(7) ) )
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 +
sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```

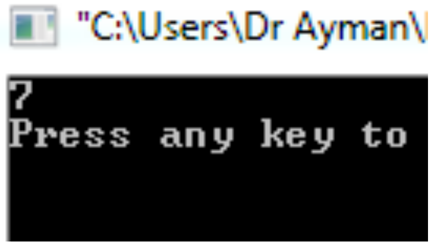

Functions
predefined
user-defined

➢Functions Scope Variables

➢ Function parameters, as well as variables defined inside the function body, are called local variables



```
#include <iostream.h>
int add(int x, int y) // function parameters x and y are local variables
{
    return x+y;
}
int main()
{
    cout<<add(3,4)<<endl;
    return 0;
}
```



"C:\Users\Dr Ayman\

```
7
Press any key to
```

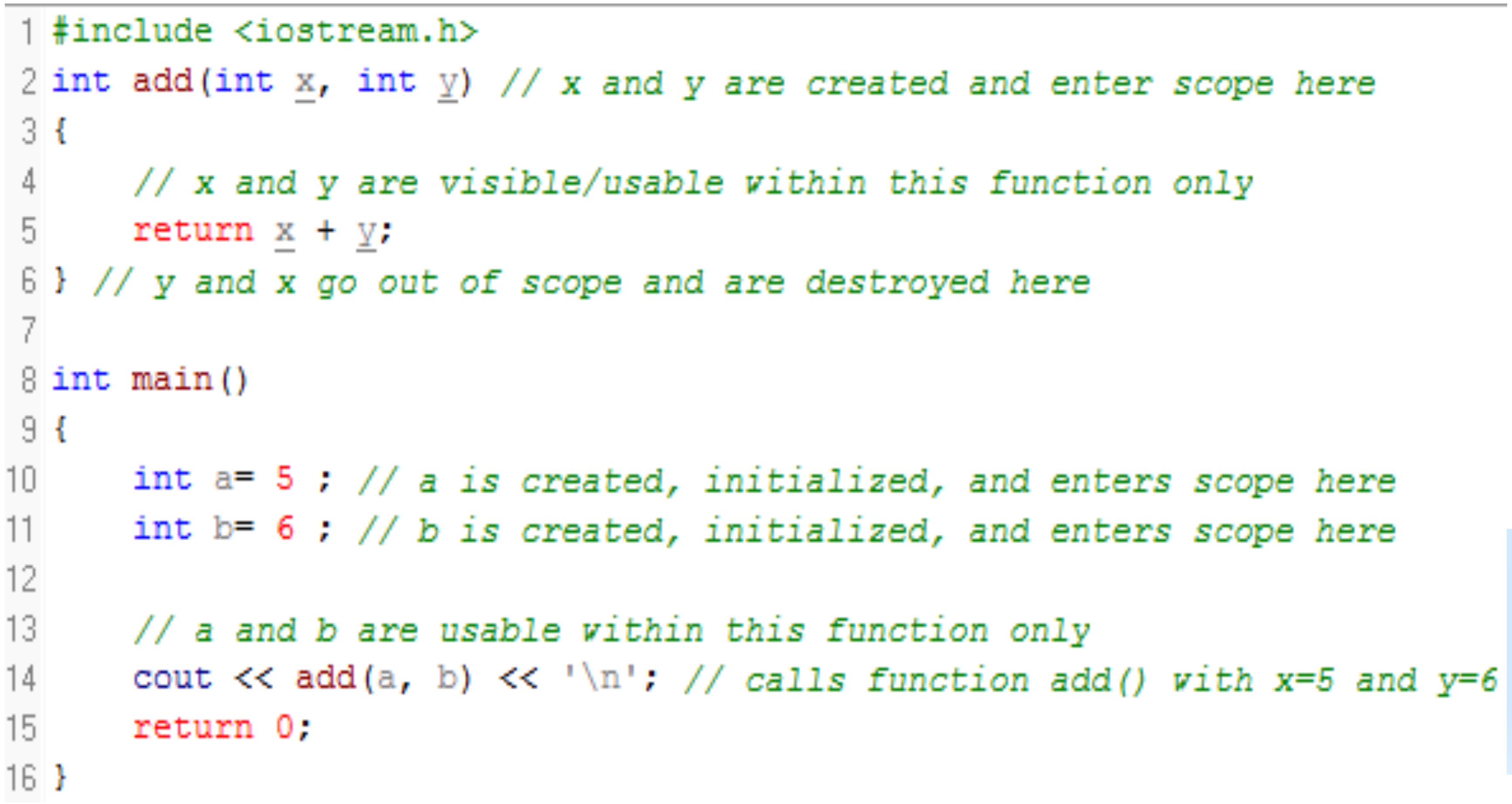
In this lesson, we'll look at some properties of local variables in more detail

➢Local variable lifetime

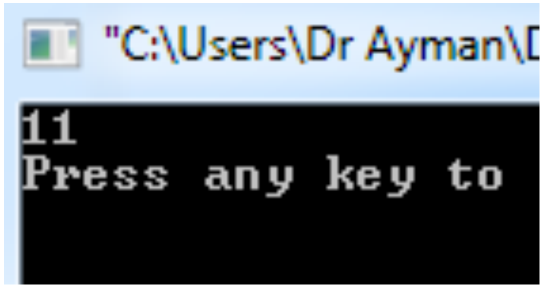
```
int add(int x, int y)             // function parameters x and y are initialized
{
   return x+y;
}                                 //  x and y destroyed here
```

➢ Much like a person’s lifetime is defined to be the time between their birth and death, an object’s lifetime is defined to be the time between its creation and destruction.

➢Local scope



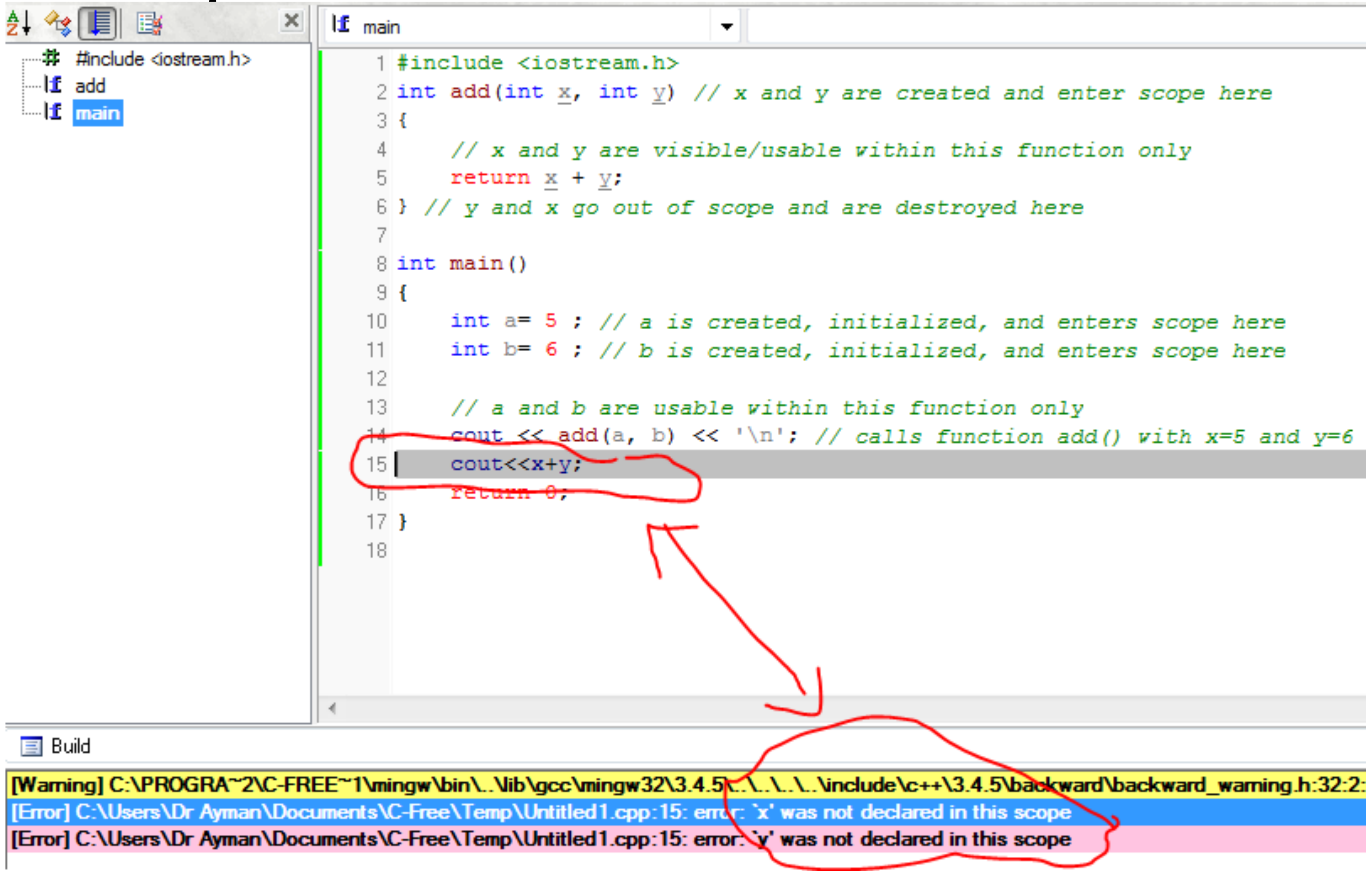
```
#include <iostream.h>
int add(int x, int y) // x and y are created and enter scope here
{
    // x and y are visible/usable within this function only
    return x + y;
} // y and x go out of scope and are destroyed here

int main()
{
    int a= 5 ; // a is created, initialized, and enters scope here
    int b= 6 ; // b is created, initialized, and enters scope here

    // a and b are usable within this function only
    cout << add(a, b) << '\n'; // calls function add() with x=5 and y=6
    return 0;
}
```



"C:\Users\Dr Ayman\

```
11
Press any key to
```

➢Local scope



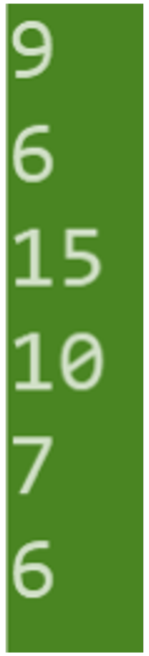
```
#include <iostream.h>
int add(int x, int y) // x and y are created and enter scope here
{
    // x and y are visible/usable within this function only
    return x + y;
} // y and x go out of scope and are destroyed here

int main()
{
    int a= 5 ; // a is created, initialized, and enters scope here
    int b= 6 ; // b is created, initialized, and enters scope here

    // a and b are usable within this function only
    cout << add(a, b) << '\n'; // calls function add() with x=5 and y=6
    cout<<x+y;
    return 0;
}
```

```
[Warning] C:\PROGRA~2\C-FREE~1\mingw\bin\..\lib\gcc\mingw32\3.4.5\..\..\..\..\include\c++\3.4.5\backward\backward_warning.h:32:2:
[Error] C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled1.cpp:15: error: `x' was not declared in this scope
[Error] C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled1.cpp:15: error: `y' was not declared in this scope
```

➢Example

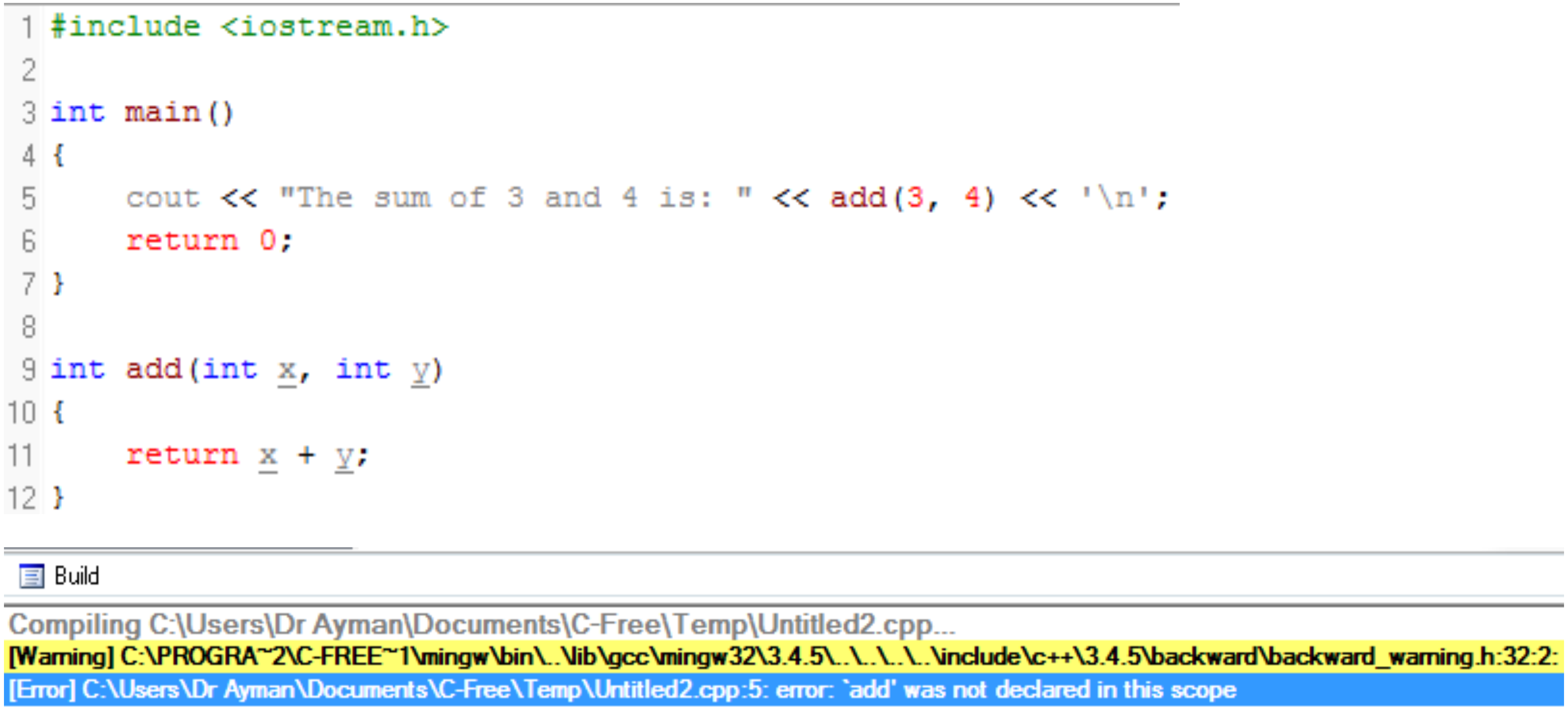
```
#include <iostream>

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

int multiply(int z, int w)
{
    return z * w;
}
```

```
int main()
{
    cout << add(4, 5) << endl;
    cout << multiply(2, 3) << endl;
    cout << add(1 + 2, 3 * 4) << endl;

     int a = 5;
    cout << add(a, a) << std::endl;
    cout << add(1, multiply(2, 3)) << endl;
    cout << add(1, add(2, 3)) << endl;
    return 0;
}
```



```
9
6
15
10
7
6
```

➢Forward declarations and definitions



```
#include <iostream.h>

int main()
{
    cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
    return 0;
}

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

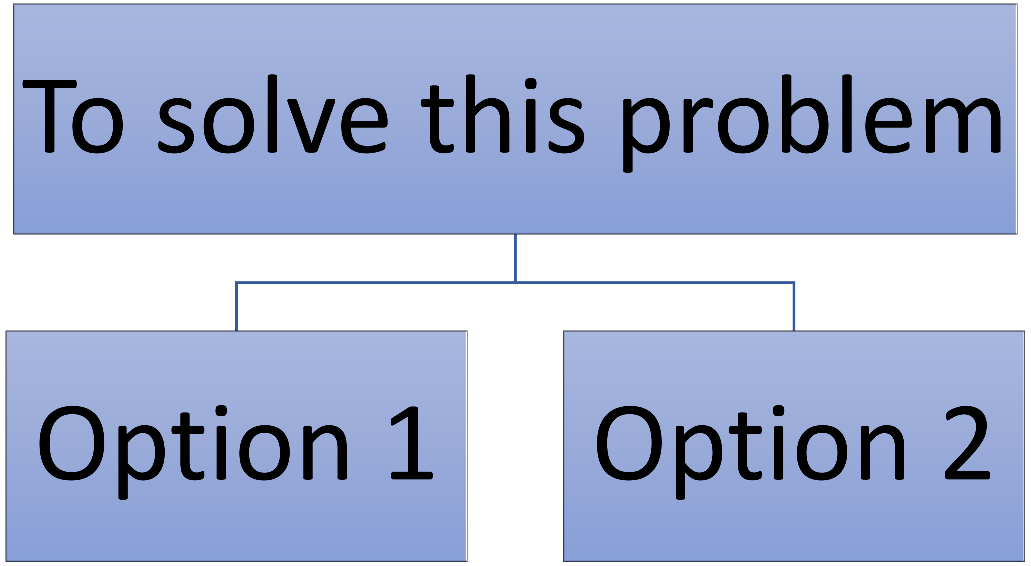
Build

```
Compiling C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled2.cpp...
[Warning] C:\PROGRA~2\C-FREE~1\mingw\bin\..\lib\gcc\mingw32\3.4.5\..\..\..\..\include\c++\3.4.5\backward\backward_warning.h:32:2:
[Error] C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled2.cpp:5: error: `add' was not declared in this scope
```

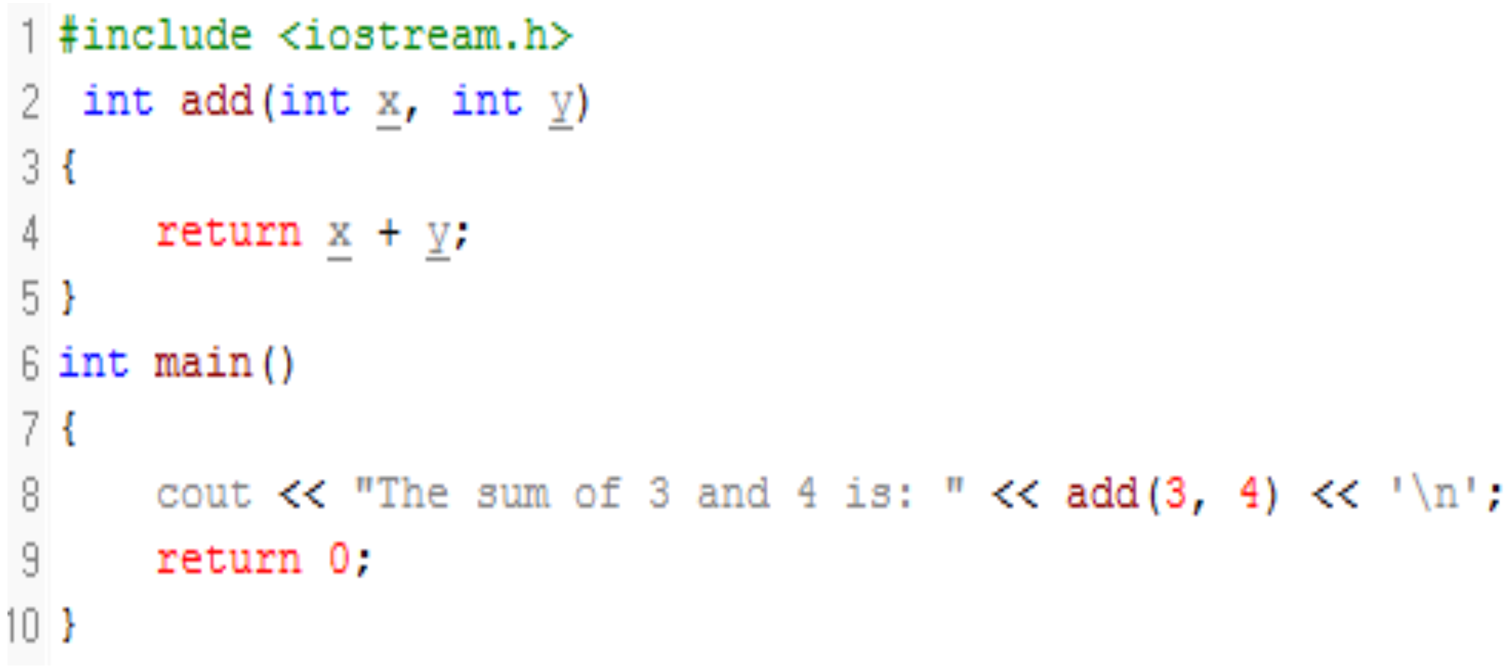
➢Forward declarations and definitions (cont.)

In fact, it doesn't compile at all! Visual Studio produces an error:

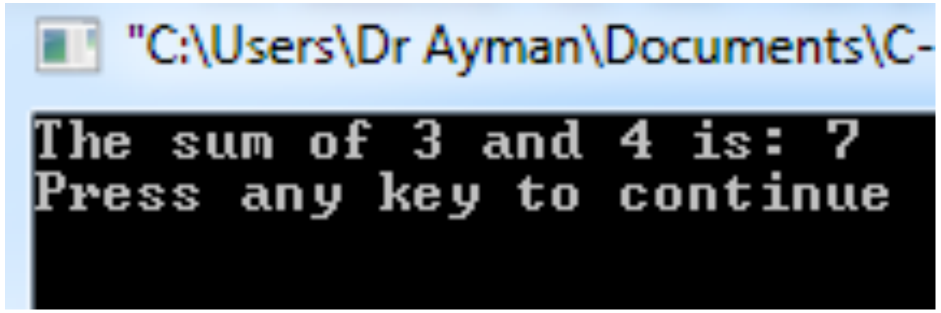
- ➢ The reason this program doesn't compile is because the compiler compiles the contents of code files sequentially. When the compiler reaches the function call to add on line 5 of main, it doesn't know what add is, because we haven't defined add until line 9! That produces the error, identifier not found.



➢Option 1: Reorder the function definitions



```
#include <iostream.h>
 int add(int x, int y)
{
    return x + y;
}
int main()
{
    cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
    return 0;
}
```

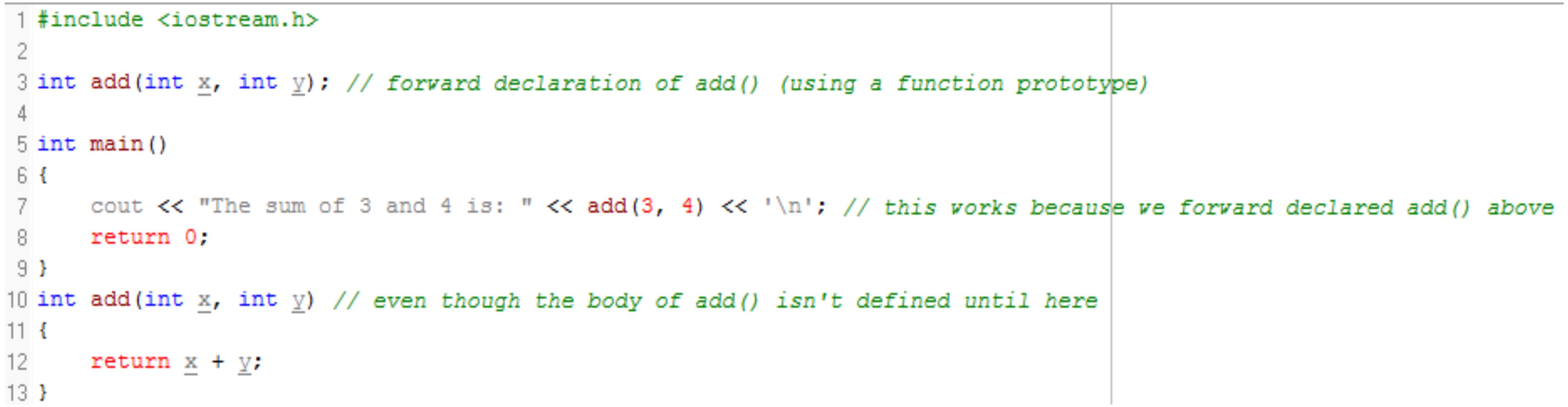


"C:\Users\Dr Ayman\Documents\C-

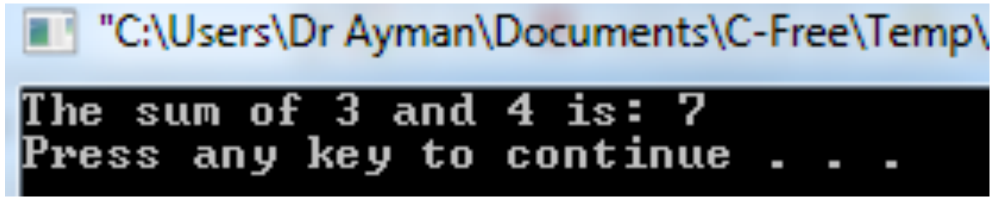
```
The sum of 3 and 4 is: 7
Press any key to continue
```

➢Option 2: Use a forward declaration

❑ A forward declaration allows us to tell the compiler about the existence of an identifier before defining the identifier.



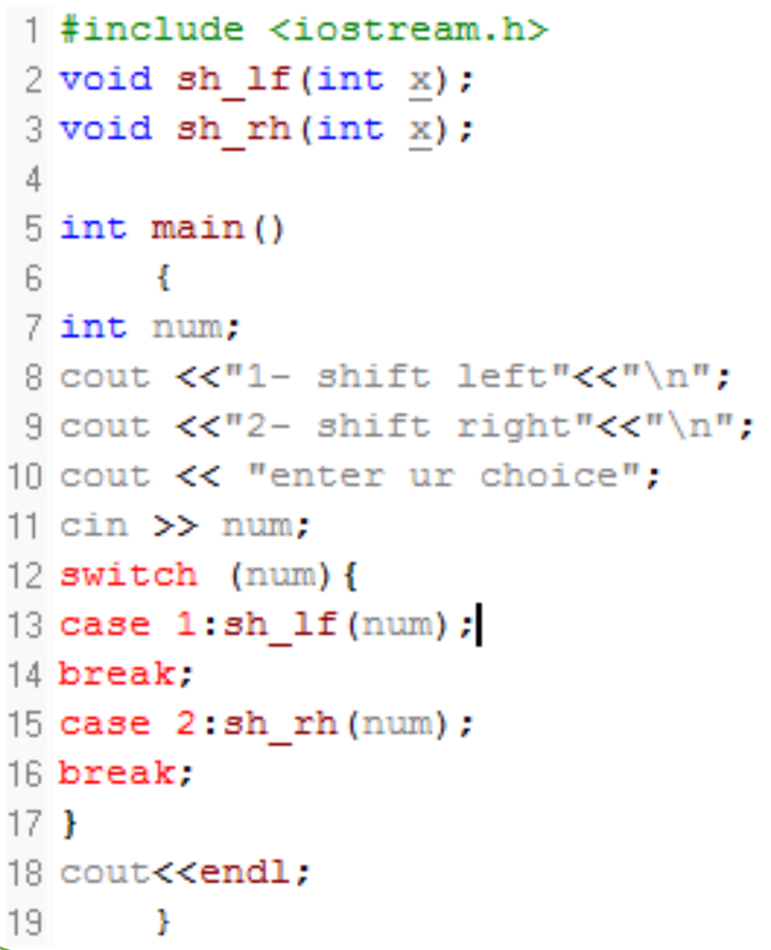
```
#include <iostream.h>

int add(int x, int y); // forward declaration of add() (using a function prototype)

int main()
{
    cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n'; // this works because we forward declared add() above
    return 0;
}
int add(int x, int y) // even though the body of add() isn't defined until here
{
    return x + y;
}
```



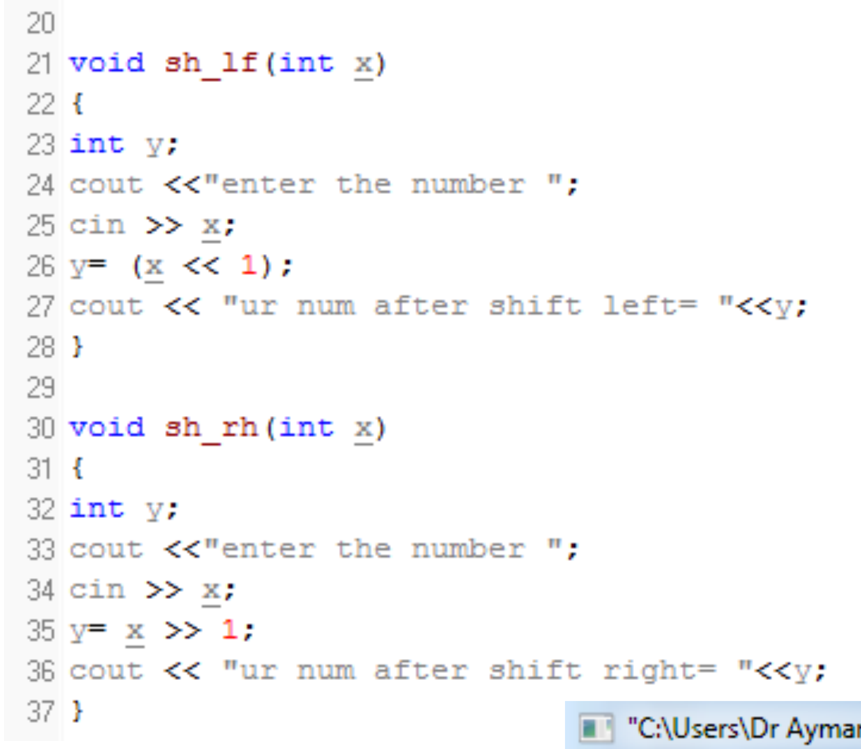
"C:\Users\Dr Ayman\Documents\C-Free\Temp\

```
The sum of 3 and 4 is: 7
Press any key to continue . . .
```

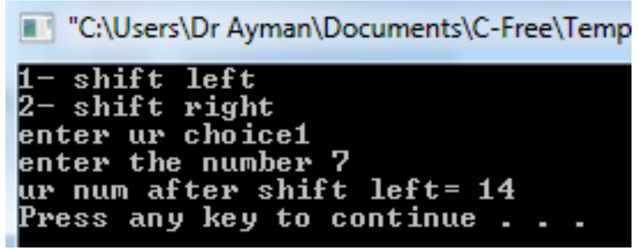
➢Example Write the program that ask the user to print shift left or shift right to any number depending on your choice.



```cpp
#include <iostream.h>
void sh_lf(int x);
void sh_rh(int x);

int main()
     {
int num;
cout <<"1- shift left"<<"\n";
cout <<"2- shift right"<<"\n";
cout << "enter ur choice";
cin >> num;
switch (num){
case 1:sh_lf(num);
break;
case 2:sh_rh(num);
break;
}
cout<<endl;
     }
```



```cpp

void sh_lf(int x)
{
int y;
cout <<"enter the number ";
cin >> x;
y= (x << 1);
cout << "ur num after shift left= "<<y;
}

void sh_rh(int x)
{
int y;
cout <<"enter the number ";
cin >> x;
y= x >> 1;
cout << "ur num after shift right= "<<y;
}
```



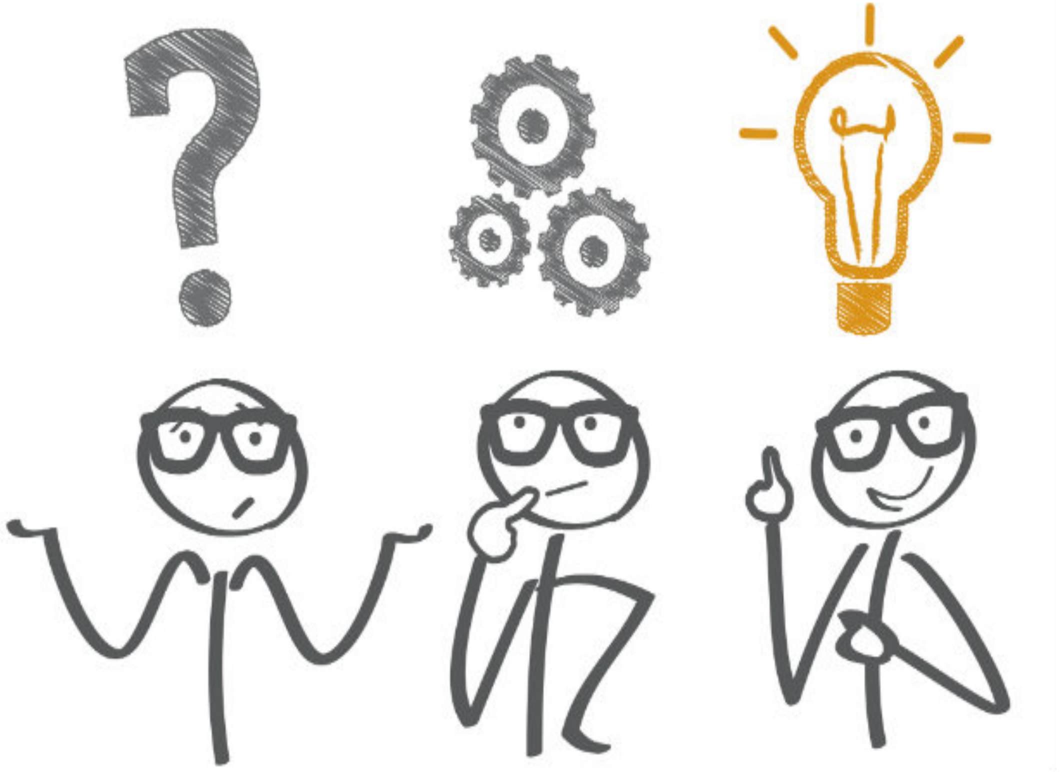
"C:\Users\Dr Ayman\Documents\C-Free\Temp

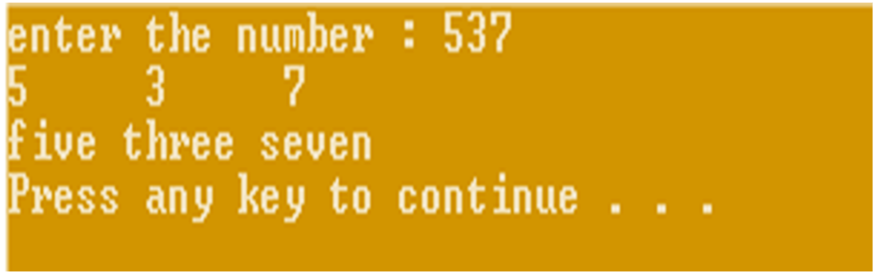
```
1- shift left
2- shift right
enter ur choice1
enter the number 7
ur num after shift left= 14
Press any key to continue . . .
```

Quiz

➢ **quiz**

Write a C ++ program to enter a three numbers and print each number individually and in words using switch case method.

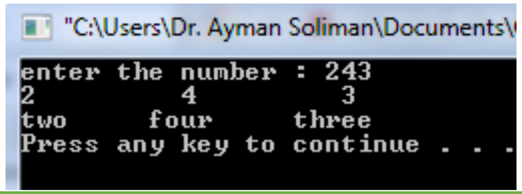




```
enter the number : 537
5     3     7
five three seven
Press any key to continue . . .
```

➢ **quiz** - solution

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
        int x,a1,a2,b1,c1;
        string a11,b11,c11;
        cout<<"enter the number : ";
        cin>>x;
        a1=x/100;
        a2=x%100;
        b1=a2/10;
        c1=a2%10;
        cout<<a1<<"    "<<b1<<"    "<<c1<<endl;
```


"C:\Users\Dr. Ayman Soliman\Documents\

```
enter the number : 243
2         4         3
two       four      three
Press any key to continue . . .
```

```
switch (a1)
{case 0:
 a11="zero";
 break;
 case 1:
 a11="one";
 break;
 case 2:
 a11="two";
 break;
 case 3:
 a11="three";
 break;
 case 4:
 a11="four";
 break;
 case 5:
 a11="five";
 break;
 case 6:
 a11="six";
 break;
 case 7:
 a11="seven";
 break;
 case 8:
 a11="eight";
 break;
 case 9:
 a11="nine";
 break;          }
```

```
switch (b1)
{case 0:
 b11="zero";
 break;
 case 1:
 b11="one";
 break;
 case 2:
 b11="two";
 break;
 case 3:
 b11="three";
 break;
 case 4:
 b11="four";
 break;
 case 5:
 b11="five";
 break;
 case 6:
 b11="six";
 break;
 case 7:
 b11="seven";
 break;
 case 8:
 b11="eight";
 break;
 case 9:
 b11="nine";
 break;          }
```

```
switch (c1)
{case 0:
 c11="zero";
 break;
 case 1:
 c11="one";
 break;
 case 2:
 c11="two";
 break;
 case 3:
 c11="three";
 break;
 case 4:
 c11="four";
 break;
 case 5:
 c11="five";
 break;
 case 6:
 c11="six";
 break;
 case 7:
 c11="seven";
 break;
 case 8:
 c11="eight";
 break;
 case 9:
 c11="nine";
 break;          }
```

```
cout<<a11<<"    "<<b11<<"    "<<c11<<endl;
return 0;
}
```

Thank you