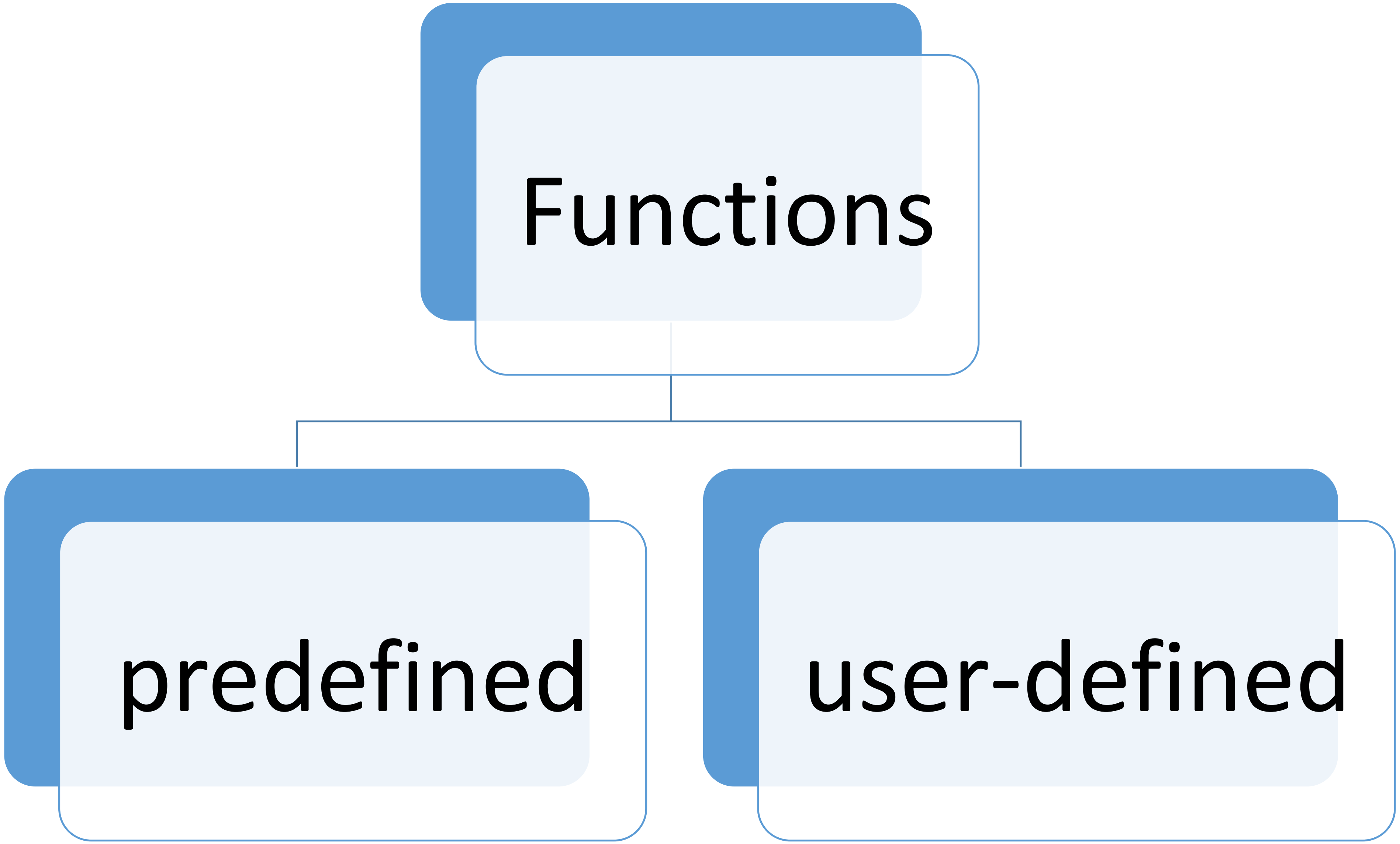
E1123 Computer Programming (a)

(Fall 2020)

Functions (2)

INSTRUCTOR

DR / AYMAN SOLIMAN


Functions
predefined
user-defined

➢Why functions are useful

- ➢ Organization → As programs grow in complexity, having all the code live inside the main() function becomes increasingly complicated. A function is almost like a mini-program that we can write separately from the main program, without having to think about the rest of the program while we write it. This allows us to reduce a complicated program into smaller, more manageable chunks, which reduces the overall complexity of our program.

- ➢ Reusability → Once a function is written, it can be called multiple times from within the program. This avoids duplicated code ("Don't Repeat Yourself") and minimizes the probability of copy/paste errors. Functions can also be shared with other programs, reducing the amount of code that must be written from scratch (and retested) each time.

➢Why functions are useful

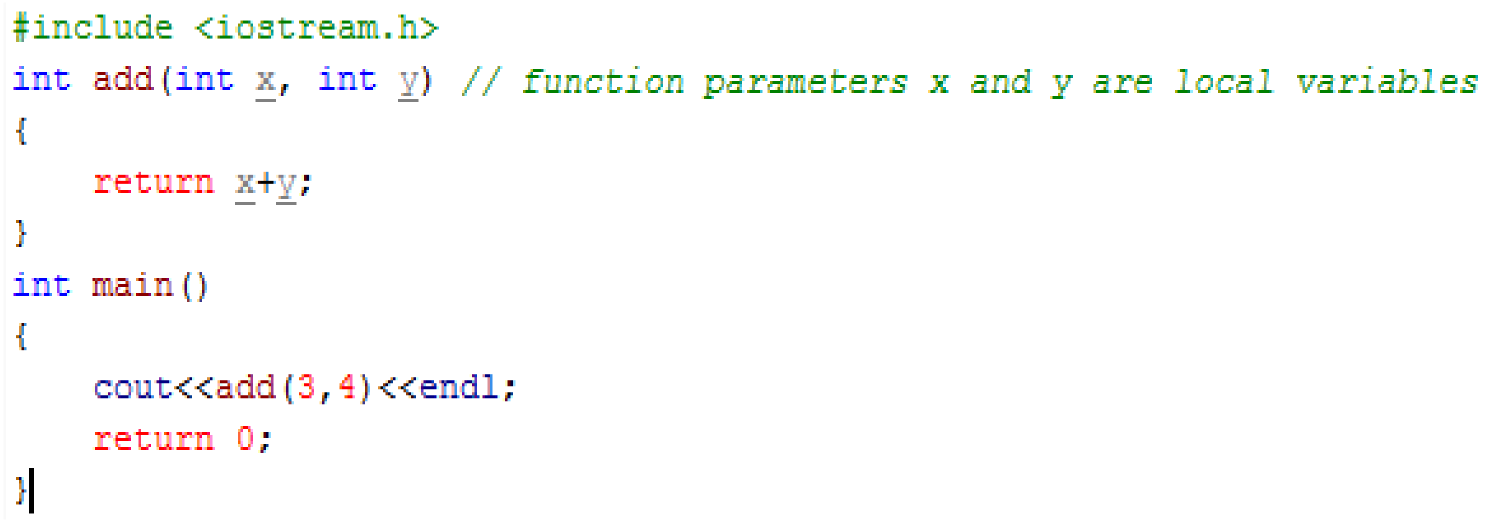
- ➢ Testing → Because functions reduce code redundancy, there’s less code to test in the first place. Also, because functions are self-contained, once we’ve tested a function to ensure it works, we don’t need to test it again unless we change it.

- ➢ Extensibility → When we need to extend our program to handle a case it didn’t handle before; functions allow us to make the change in one place and have that change take effect every time the function is called.

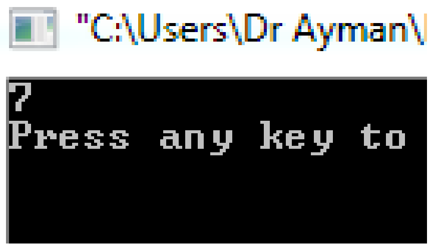
- ➢ Abstraction → In order to use a function, you only need to know its name, inputs, outputs, and where it lives. You don’t need to know how it works, or what other code it’s dependent upon to use it. This lowers the amount of knowledge required to use other people’s code (including everything in the standard library).

➢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
```

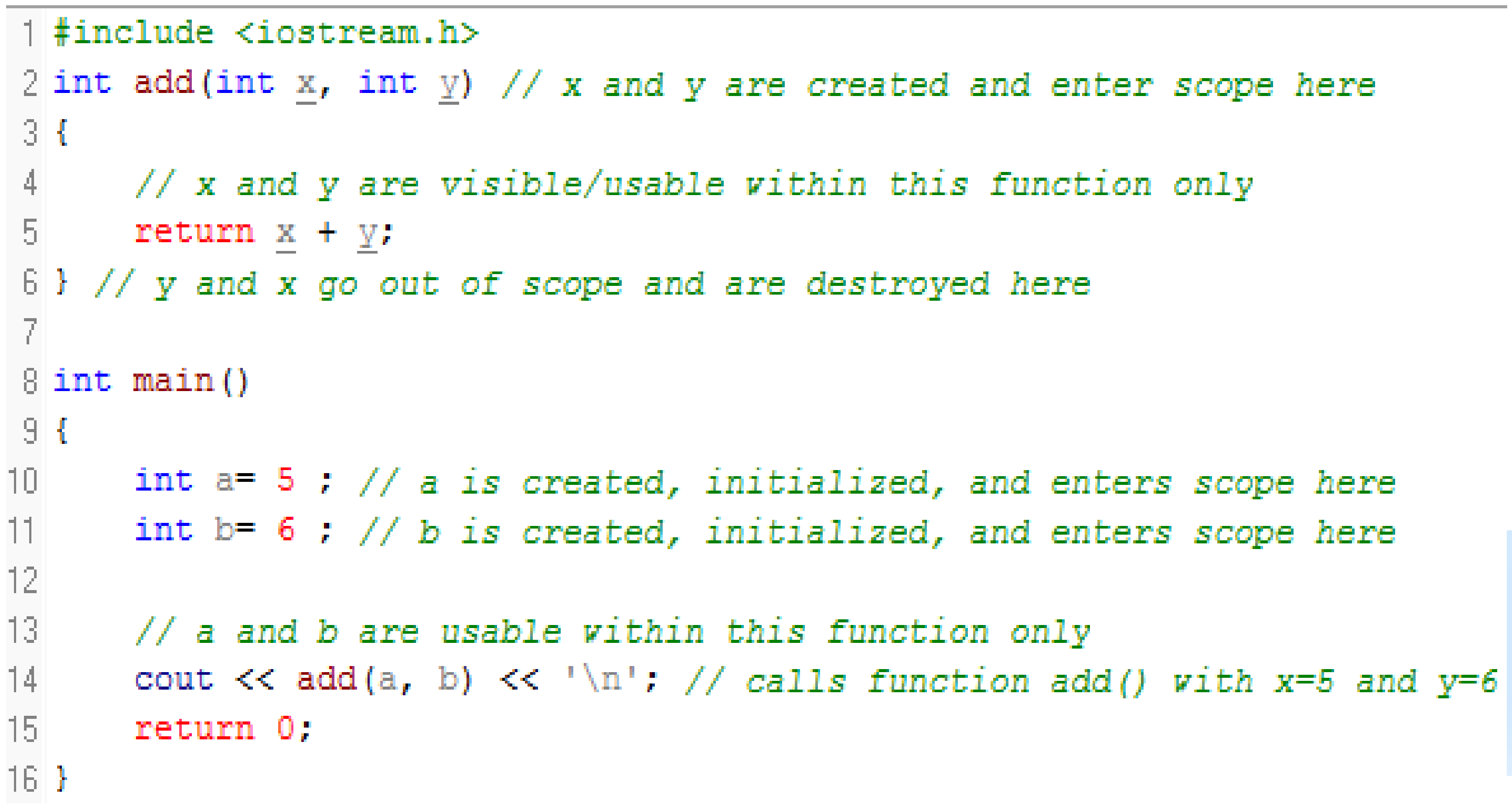
In this lesson, we'll look at some properties of local variables in more detail

➢Local variable lifetime

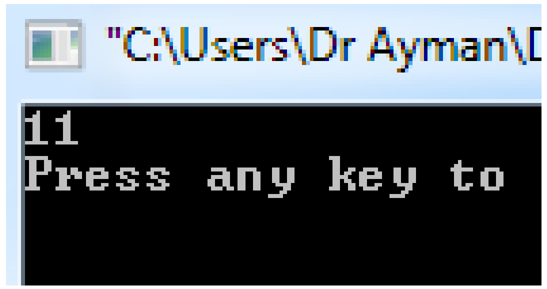
```
int add(int x, int y)                // function parameters x and y are initialized
{
    return x+y;
}                                    //  x and y destroyed here
```

➢ 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



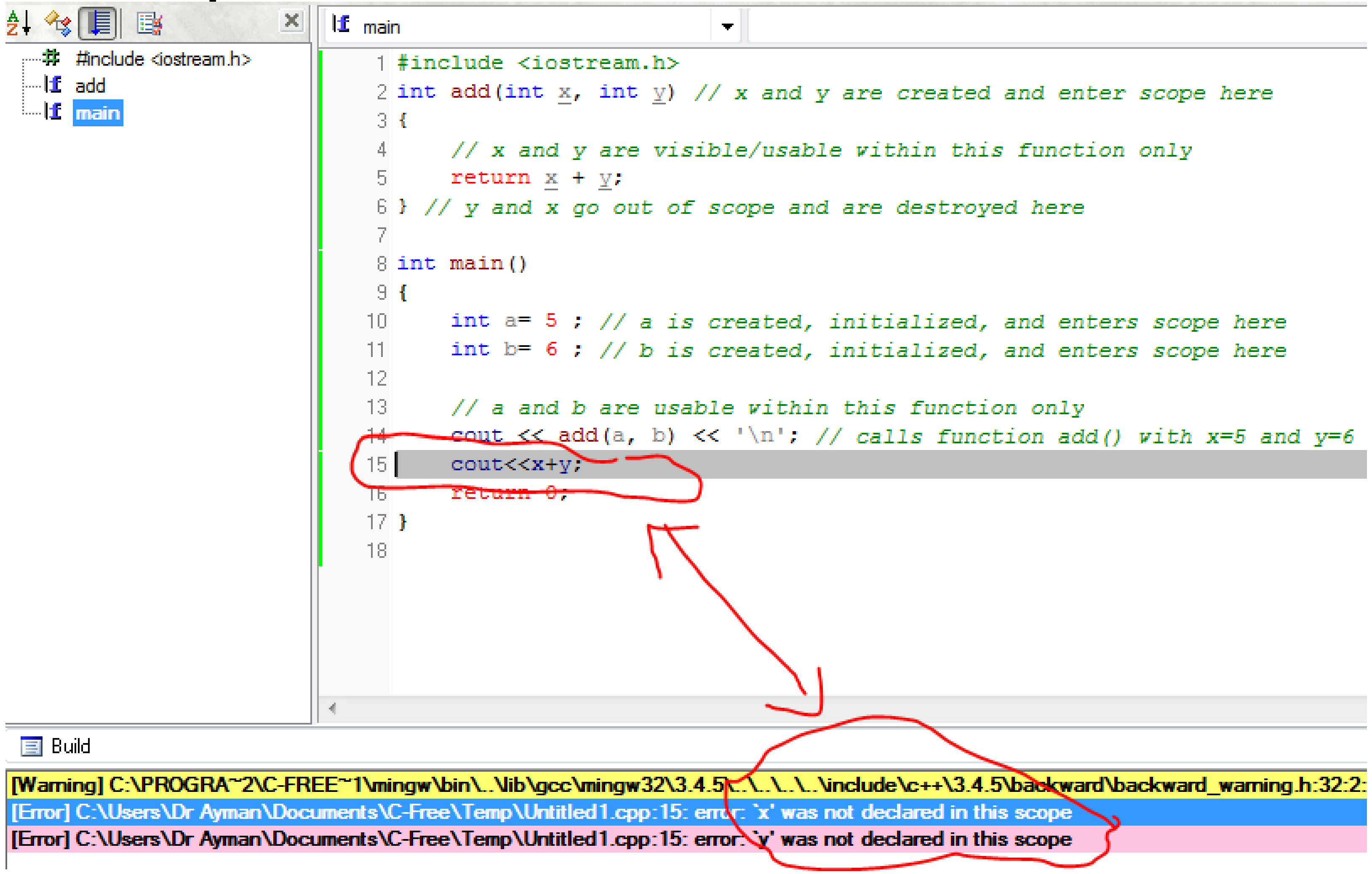
```cpp
#include <iostream.h>
int add(int x, int y) // x and y are created and enter scope here
{
    // x and y are visible/usable within this function only
    return x + y;
} // y and x go out of scope and are destroyed here

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\D

```
11
Press any key to
```

➢Local scope



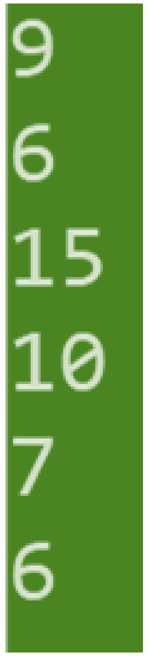
```
#include <iostream.h>
int add(int x, int y) // x and y are created and enter scope here
{
    // x and y are visible/usable within this function only
    return x + y;
} // y and x go out of scope and are destroyed here

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

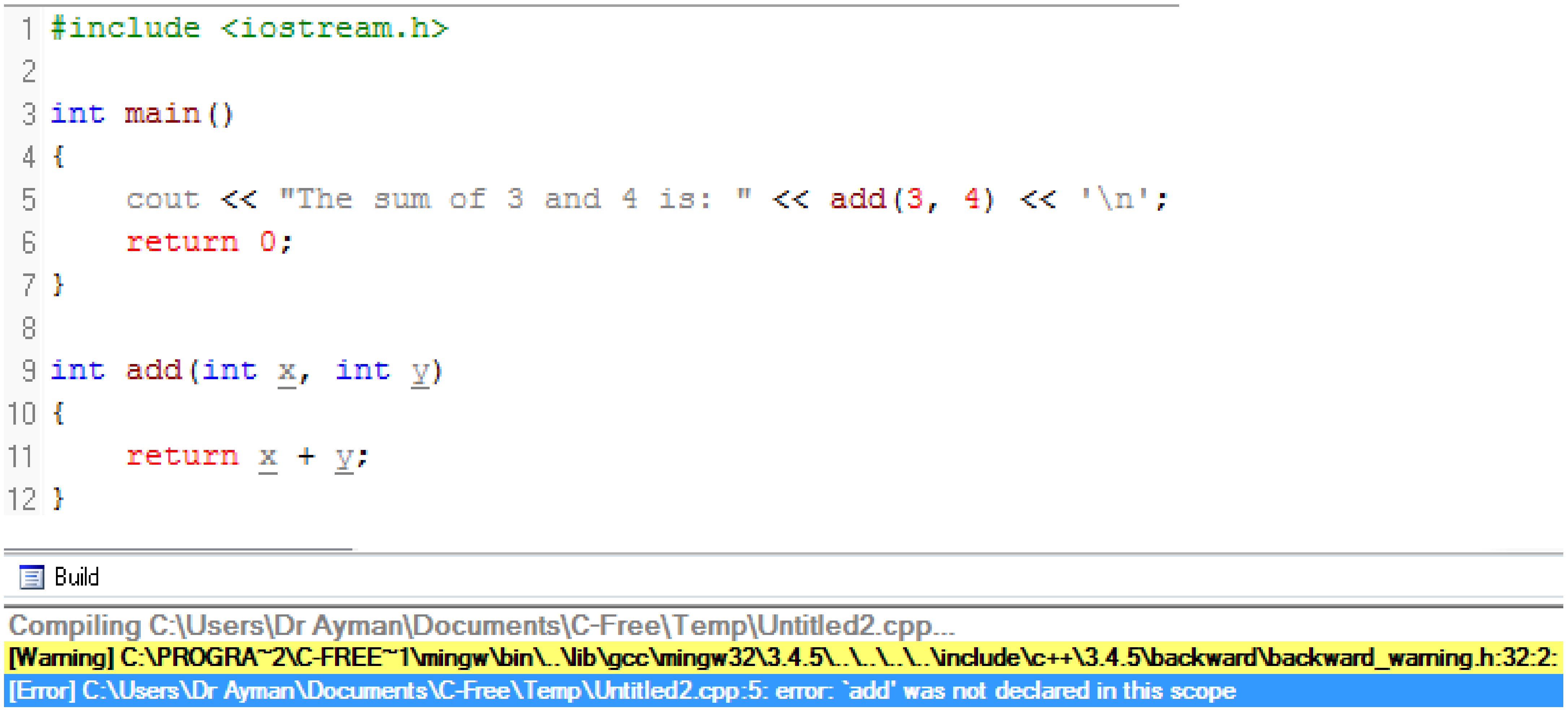
```
#include <iostream>

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;
}
```

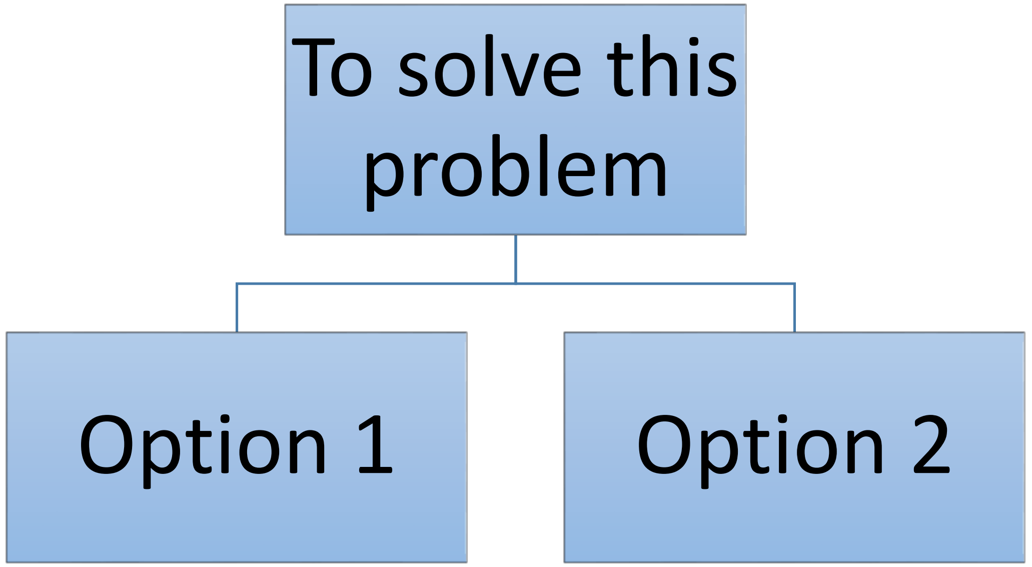
Build

```
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
```

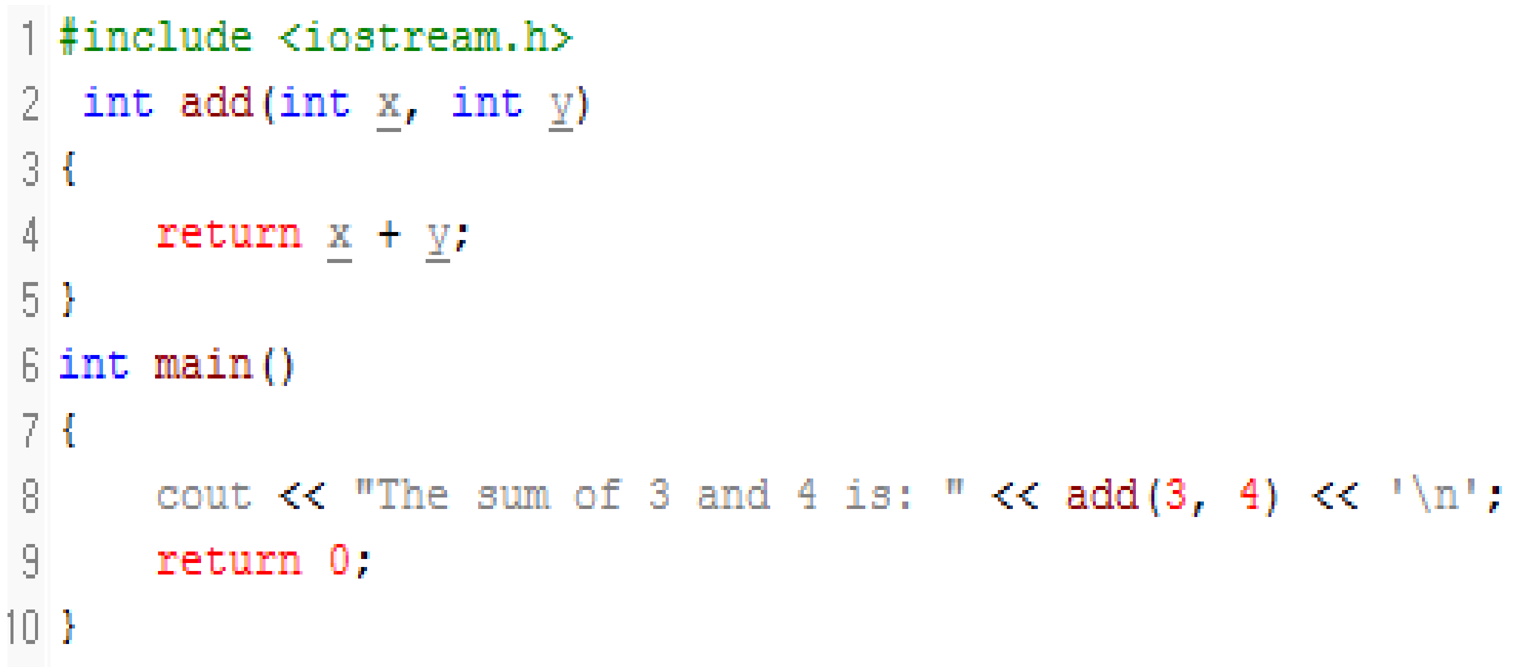
➢Forward declarations and definitions (cont.)

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

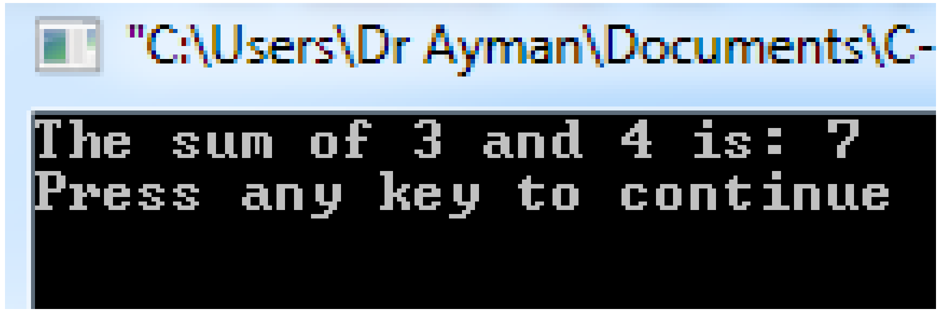
- ➢ The reason this program doesn't compile is because the compiler compiles the contents of code files sequentially. When the compiler reaches the function call to add on line 5 of main, it doesn't know what add is, because we haven't defined add until line 9! That produces the error, identifier not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



➢Option 1: Reorder the function definitions



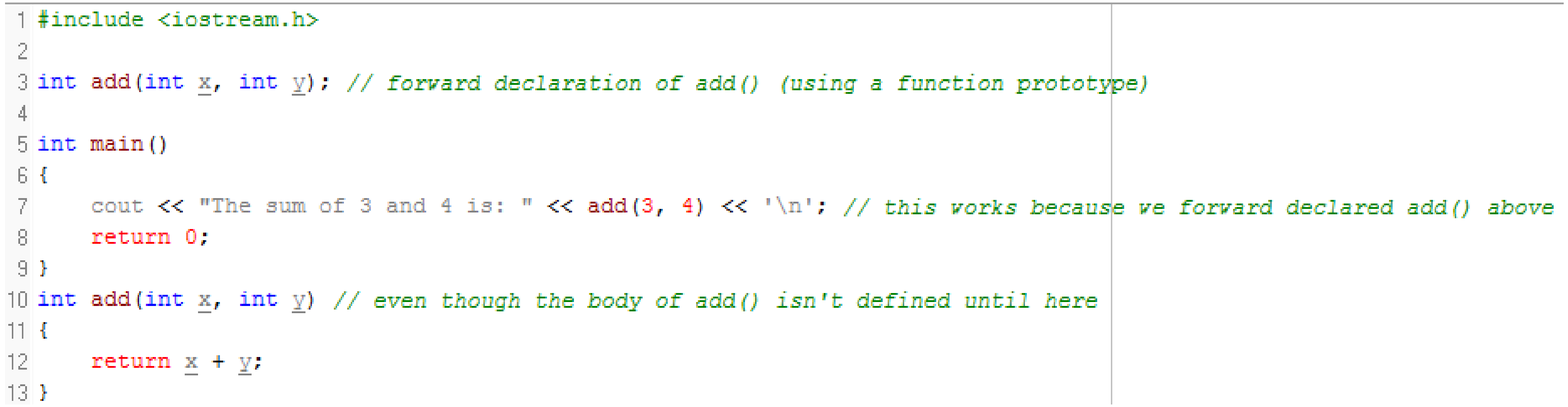
```
#include <iostream.h>
 int add(int x, int y)
{
    return x + y;
}
int main()
{
    cout << "The sum of 3 and 4 is: " << add(3, 4) << '\n';
    return 0;
}
```



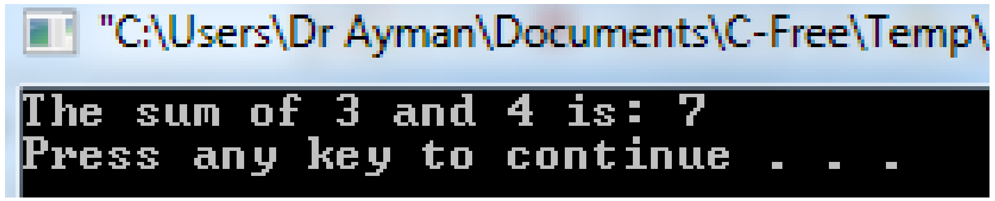
```
The sum of 3 and 4 is: 7
Press any key to continue
```

➢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.

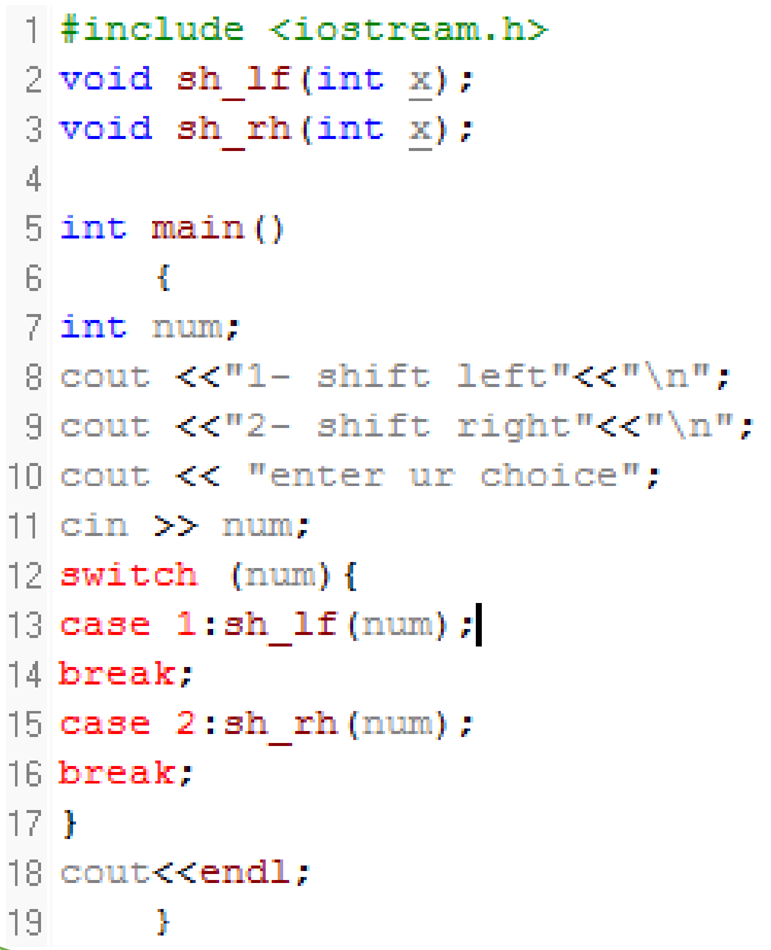


```cpp
#include <iostream.h>

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;
}
```

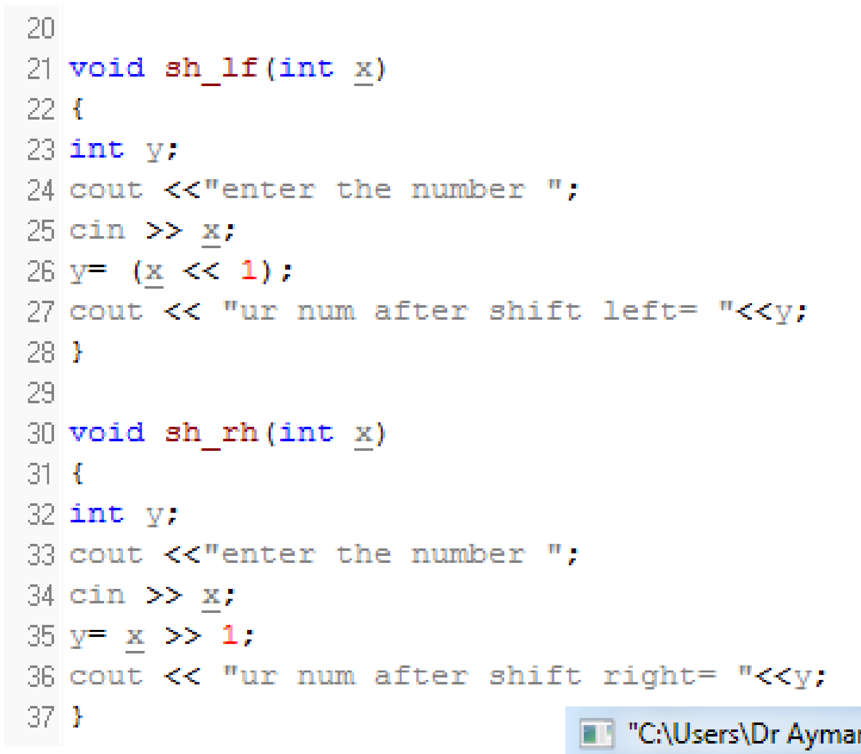


```
"C:\Users\Dr Ayman\Documents\C-Free\Temp\
The sum of 3 and 4 is: 7
Press any key to continue . . .
```

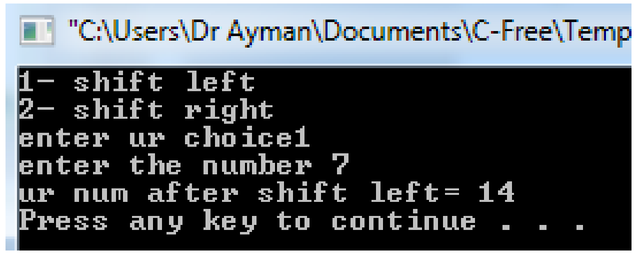
➢Example Write the program that ask the user to print shift left or shift right to any number depending on your choice.



```
#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;
     }
```



```

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 . . .
```

Thank
you