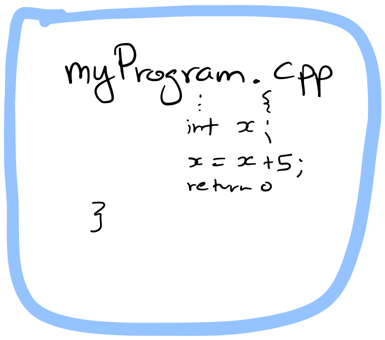
C++ Basics – Standard I/O and namespaces

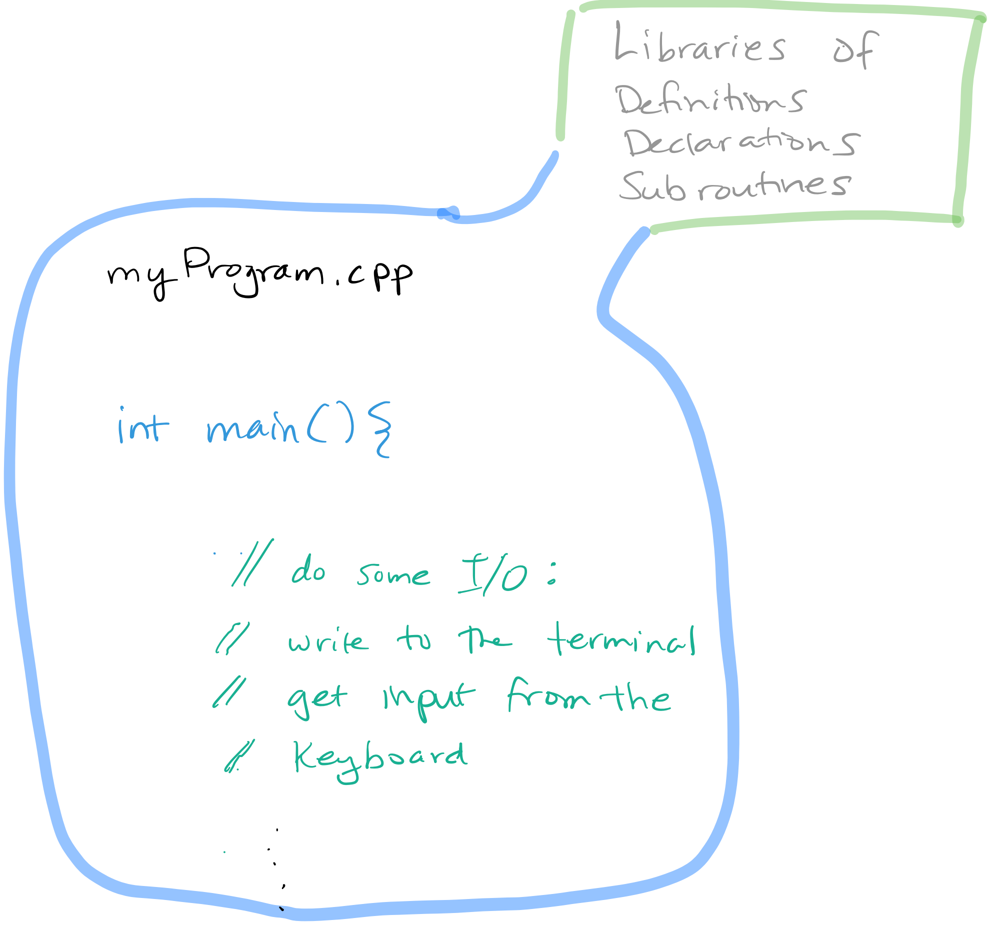


myProgram.cpp

```
    :     {
    int x;
    x = x+5;
    return 0
}
```

This program is self contained, does no Input/Output (I/O) and calls no external subroutines – just compile it and run it.

C++ Basics – Standard I/O and namespaces



```
int main(){

    // do some I/O:
    // write to the terminal
    // get input from the
    // keyboard
        .
        .
        .
```

C++ Basics – Standard I/O and namespaces

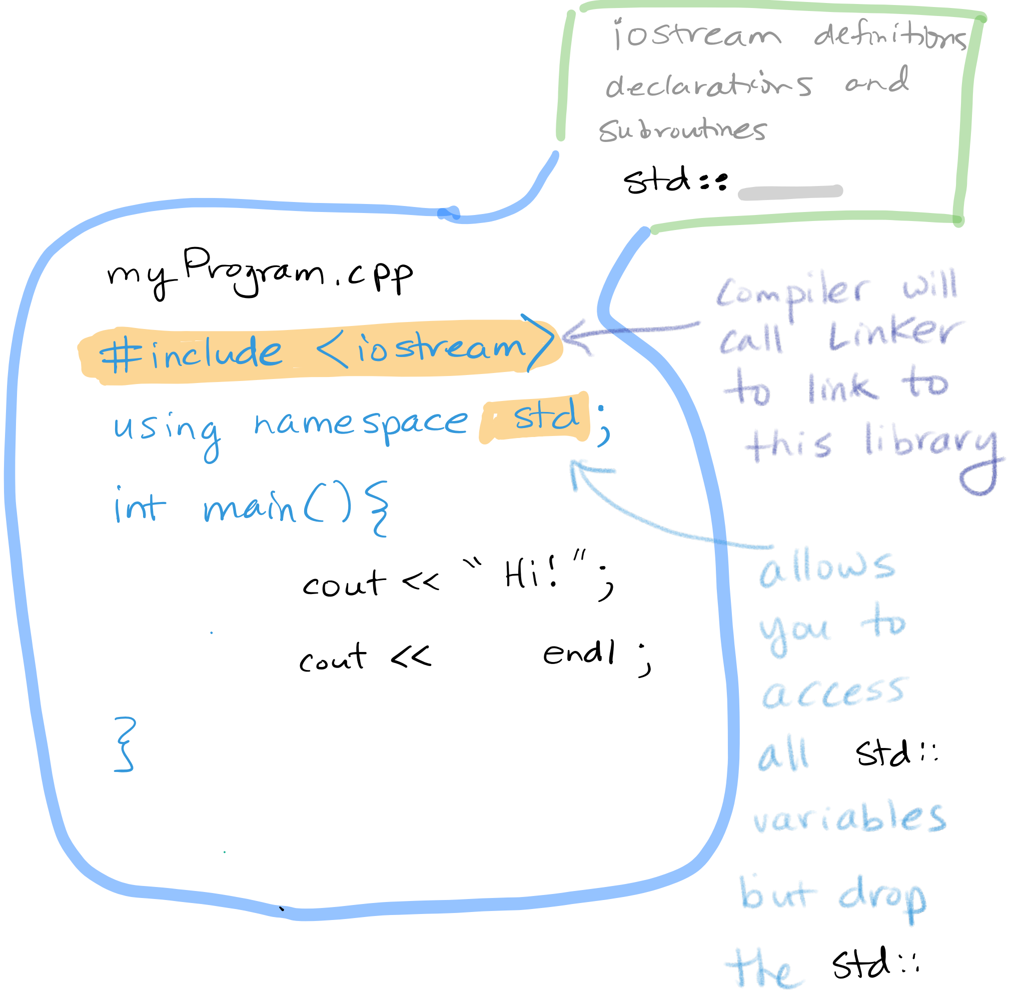
iostream definitions declarations and subroutines

std::

myProgram.cpp

```
#include <iostream>
int main(){
        std::cout << "Hi!";
        std::cout << std::endl;
}
```

C++ Basics — Standard I/O and namespaces



iostream definitions declarations and subroutines

std:: ____

```
myProgram.cpp
#include <iostream>
using namespace std;
int main(){
        cout << " Hi! ";
        cout <<      endl;
}
```

Compiler will call Linker to link this library

allows you to access all std:: variables but drop the std::

What if you name a variable with the same name as is used in std? Problems arise.
That's why serious coders usually don't use a namespace; they just use the prefix.

For brevity, we may write "code fragments"

```
cout << "What's your favourite bunny?
    Enter an integer: ";
cin >> favourite;
cout << "I love bunny #" <<
    favourite << ", too! " << endl;
```

We will assume such fragments are properly prefaced with "#include <iostream> using namespace std"