Program organisation and Functions

- You should always either use-

```
#include<iostream>

using namespace std;
int main(){
  cout<<"LMAO"<<endl;
  int n;
  cin>>n;
}
```

Or you will have to write-

```
#include<iostream>

int main(){
  std::cout<<"LMAO"<<std::endl;
  int n;
  std::cin>>n;
}
```

- If you have a public, well established header file like iostream or cmath, write it #include<iostream> but if it is something you have defined, you will have to write #include “bullet.h”. (Note- Some header files have to be written with .h, some with .hpp and some without anything as a postscript)

- If your program has a specific function that is defined but isn’t declared, then you need to link other function along with it

```
1   #include<iostream>
2
3   using namespace std;
4
5   int a(int, int);
6
7   int main(){
8       cout<<"Bolo shivaji"<<endl;
9       cout<<a(3,5);
10      return 0;
11  }
12
```

The above code gives an error because we didn’t include any definition for a(). We just declared it.

- We can use the header files to not have to declare a everywhere in case we like to use it a lot. We can also define a inside header files so that is that.

Untitled3.cpp | header1.h | shooter.h | Mr. Checka.cpp

```cpp
#include<iostream>
#include "header1.h"
using namespace std;

int a(int, int);

int main(){
    cout<<"Bolo shivaji"<<endl;
    cout<<a(3,5);
    return 0;
}

```

Untitled3.cpp | header1.h | shooter.h | Mr. Checka.cpp

```cpp
int a(int b, int c){
    return b+c;
}

```

Just make sure to save all files in the same folder else Untitled3.cpp won't be able to find header1.h at all

(Fun fact- We did not even need to declare int a(int, int) in there. I just put it in show that declaring twice doesn't give an error **but defining twice will give an error**)

- Look at this my man-

Untitled3.cpp

```cpp
#include<iostream>
#include "header1.h"
using namespace std;

int main(){
    cout<<"Bolo shivaji"<<endl;
    cout<<bhrat::a(3,5)<<endl;
    cout<<a(3,5);
    return 0;
}
```

header1.h | shooter.h | Mr. Checka.cpp

```cpp

int a(int b, int c){
    return (b>c)?(b-c):(c-b);
}
namespace bhrat{
    int a(int b, int c){
        return b+c;
    }
}
```

```
"C:\Users\aryav\Desktop\IIT-B\CS101\Rough Files\Untitled3.exe"
Bolo shivaji
8
2
Process returned 0 (0x0)   execution time : 0.359 s
Press any key to continue.
```

This clearly shows the following things-

1. how to use namespaces and how global/std namespace is the default one
2. ternery operators just because I want to show off

BTW you can use #include<simplecpp> or #include<iostream> inside header files as well

- A better feel for namespaces and stuff-

header1.h | Untitled3.cpp

```cpp
#include<iostream>
#include "header1.h"
using namespace std;

void bruh(){
    cout<<"bruh"<<endl;
}

int bhrat::bruh(int a, int b){
    cout<<"Tis just a void lmao"<<endl;
    return 21;
}

int main(){
    bruh();
    cout<<bhrat::bruh(2,3)<<endl;
    cout<<"Bolo shivaji"<<endl;
    cout<<bhrat::a(3,5)<<endl;
    cout<<a(3,5);
    return 0;
}
```

```
"C:\Users\aryav\Desktop\IIT-B\CS101\Rough Files\Untitled3.exe"
bruh
Tis just a void lmao
21
Bolo shivaji
8
2
Process returned 0 (0x0)   execution time : 0.803 s
Press any key to continue.
```

header1.h | Untitled3.cpp

```
1
2   int a(int b, int c){
3       return (b>c)?(b-c):(c-b);
4   }
5
6   void bruh();
7
8   namespace bhrat{
9       int a(int b, int c){
10          return b+c;
11      }
12
13      int bruh(int, int);
14  }
15
```

Fun fact- You can put “using namespace anywhere in the code. It can even be inside function definitions/bodies

- Fun fact that will put you in existential crisis- if you write- WARNING THE BELOW CODE GIVES ERROR DUE TO AMBIGUITY

```
#include<iostream>
using namespace std;
namespace N{
  void cout(int a){
    while(a>0){
      cout<<a%10;
      a/=10;
    }
  }
}

int main(){
  cout<<12;
  using namespace N;
  cout(12);
  return 0;
}
```

Thus it is best not to use ‘using namespace’ excessively and rather believing in namespace:: supremacy

```cpp
#include<iostream>
using namespace std;
namespace N{
    void cout(int a){
        while(a>0){
            std::cout<<(a%10);
            a/=10;
        }
    }
}

int main(){
    cout<<12;
    using namespace N;
    N::cout(12);
    return 0;
}
```

```
"C:\Users\aryav\Desktop\IIT-B\CS101\Rough Files\namespace_ke_saath
1221
Process returned 0 (0x0)   execution time : 0.356 s
Press any key to continue.
```

- Functions **we** define don’t give a fuck about “using namespace” **in some cases**. They will go with global if our namespace doesn’t contain the function we talked about. (Global refers to the default namespace)

```cpp
#include<iostream>
using namespace std;
int a(int, int);

namespace k{
    int a(int l){
        return l+1;
    }
}

int main(){
    cout<<a(3,4);
    cout<<endl<<k::a(7);
    return 0;
}

int a(int a, int b){
    return a+b;
}
```

```
"C:\Users\aryav\Desktop\IIT-B\CS101\Rough Files\Untitled6.exe"
7
8
Process returned 0 (0x0)   execution time : 0.629 s
Press any key to continue.
```

```cpp
#include<iostream>
//BTW writing "using namespace k" here will give an error
int a(int, int);

namespace k{
    int a(int l){
        return l+1;
    }
}

using namespace k;

int main(){
    std::cout<<a(3,4);
    std::cout<<std::endl<<a(7);
    return 0;
}

int a(int a, int b){
    return a+b;
}
```

```
"C:\Users\aryav\Desktop\IIT-B\CS101\Rough Files\Untitled6.exe"
7
8
Process returned 0 (0x0)   execution time : 0.815 s
Press any key to continue.
```

Here we learn 2 things-

1. Function and variable can have the same name- no worries
2. If you define a function “void yass()” and then in main call it with “yass();”, if it exists in the namespace we stated with the “using namespace” line, that yass(); gets called. Else it looks for a yass() in the global namespace.

Note- Compiler does not allow you to do something like this-

```cpp
#include<iostream>
//BTW writing "using namespace k" here will give an error
int a(int, int);

namespace k{
    int a(int l, int k){
        return l-k;
    }
}

using namespace k;

int main(){
    std::cout<<a(3,4);
    std::cout<<std::endl<<a(7);
    return 0;
}

int a(int a, int b){
    return a+b;
}
```

Because with “using namespace”, it becomes ambigious which a(int, int) you are referring to. Thus a rule of thumb-

a. Don’t use “using namespace” if you have multiple namespaces inside your program

b. If you still use “using namespace”, don’t name functions similar (compiler will not take offense tbh unless you fuck up the c. subpoint ; I am saying this for your own safety)

c. If you still name them similar, don’t give them the **same number** of formal parameters cause compiler kinda treats this as a overloading thing. (Even if they are different types, compiler will still give error of ambiguity-

```
 3  int a(int, int);
 4
 5  namespace k{
 6      int a(int l, int k){
 7          return l-k;
 8      }
 9  }
10
11  namespace yoohoo{
12      int a(int b, char c){
13          return b*((int)c);
14      }
15  }
16
17  using namespace k;
18
19  int main(){
20      std::cout<<a(3,'c');
21      std::cout<<std::endl<<a(7);
22      return 0;
23  }
24
25  int a(int a, int b){
26      return a+b;
27  }
```