Namespace, Scope, and Keywords

Presented By:

Namespace

- Some functions share their name with other functions

- Compiler has no way of differentiating them

- Namespace specifies which function you actually want to run

Example of Namespace

```
namespace name{
        void funct1(){
                //code
        }
}
name::funct1()
```

Namespace cont’d

- To specify a function every time you use it :

  (put namespace here)::(put funct here)

- To specify a namespace for a file, use using :

  using namespace (put namespace here);

- To specify a single function for a whole file :

  using (put namespace of funct here)::(put funct here);

How you will use namespace

- The only thing you will likely use namespace for :

  using namespace std;

- This tells the compiler that you are using the standard library

- This prevents you from having to type std:: before every command

Scope

- The scope of a variable is the areas in which it can be called.
- Set of brackets that directly contain it and all brackets inside that set :

```
int y = 1;
while(y==1){
        int z = y +4;
        y = y*z/5;
}
```

- In the above example, the scope of the variable z is the while loop.

Example of Scope

- Would this return an error?

```
using namespace std;
int c = 7;
while(c < 10){
        int num = c * 3;
        if (num > 20){
                c++;
        }
}
int ans = num + 1;
```

Keywords

- C++ has some words to which it has assigned properties.
- These are called keywords and help with a variety of tasks.
- You should NEVER use a keyword as a variable or function name.

Examples of Keywords

- break
- case
- const
- default
- do
- else
- enum
- for
- include
- if
- int (as well as double, float,...)
- return
- static
- switch
- void
- while
- cout
- continue