

CS 161

Intro to CS I

More About Functions:
Scope, Return Values, Default Values,
and Overloading


void Functions

- Doesn't return a value
- Still has arguments/parameters

Scope (Visibility)

- Part of program in which a declaration is valid
- Local variable
 - Declared inside a function only accessible inside function
- Localizing variables
 - Declaring variable in innermost scope

Illegal access outside loops

```
for(xint? = 0; x < 10; x++) {  
    int y = 10;  
    cout << "The value of x * y is: " << x*y << endl;   
}  
cout << "The value of y is: " << y << endl; /*y outside scope*/
```

- How do we fix this?
- What about if/else blocks?

Illegal access in functions

```
int main () {  
    int x=2, y=3;  
    compute_sum();  
    sum = x+y; //error: sum hasn't been declared  
    return 0;  
}  
  
void compute_sum() {  
    int sum = x+y; //error: x and y outside scope  
}
```

Arguments/Parameters Demo

- How could you make an `is_positive_int()` function?
- How about an `is_int()`?
- Does it make sense to have it void?

* Make a function do one thing + one thing only!

* Name the function appropriately to indicate what it does.

Making it a void function...

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
#include <string> //c++ strings
#include <cstdlib> //atoi()
4
using namespace std;
6
7 void get_positive_int(); //function declaration/prototype
8
9 int main() {
10     //not extremely useful to have void function in this case, and we should
11     //rename the function to get_positive_int, since it gets the int and checks
12     get_positive_int(); //function call
13
14     return 0;
15 }
16 //function definition
17 void get_positive_int(){
18     int x;
19     string s; //create a string object
20     bool bad; //create a flag to indicate bad or good data
21
-- INSERT --
11,79 5%
```

Making it a void function...

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
18 int x;
19 string s; //create a string object
20 bool bad; //create a flag to indicate bad or good data
21
22 do {
23     bad=false; //assume the user is not going to supply bad data
24
25     cout << "enter int: ";
26     cin >> s; //read data as a string, so it will never fail.
27     //check that all the characters are 0-9 for a postivie int
28     for(int i=0; i<s.length(); i++)
29         if(!(s.at(i)>='0' && s.at(i)<='9')) {
30             bad=true; //if a char in the string is not 0-9, not positive int
31             break; //break out of the for loop when seeing bad data
32         }
33 } while(bad); //while the user entered bad data re-prompt them
34
35 x=atoi(s.c_str());
36 cout << x << endl;
37 }
-- INSERT -- 37,2 Bot
```


Returning Values Demo...

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
1 #include <iostream>
2 #include <string> //c++ strings
3 #include <cstdlib> //atoi()
4
5 using namespace std;
6
7 int get_positive_int(); //function declaration/prototype
8
9 int main() {
10     int x; //we need a way to capture the information coming back to us
11     x=get_positive_int(); //function call
12     cout << x << endl;
13     return 0;
14 }
15 //function definition
16 int get_positive_int(){
17     int x;
18     string s; //create a string object
19     bool bad; //create a flag to indicate bad or good data
20
-- INSERT --
6,1 Top
```

Returning Values Demo...

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
18 string s; //create a string object
19 bool bad; //create a flag to indicate bad or good data
20
21 do {
22     bad=false; //assume the user is not going to supply bad data
23
24     cout << "enter int: ";
25     cin >> s; //read data as a string, so it will never fail.
26     //check that all the characters are 0-9 for a postivie int
27     for(int i=0; i<s.length(); i++)
28         if(!(s.at(i)>='0' && s.at(i)<='9')) {
29             bad=true; //if a char in the string is not 0-9, not positive int
30             break; //break out of the for loop when seeing bad data
31         }
32     } while(bad); //while the user entered bad data re-prompt them
33
34     x=atoi(s.c_str());
35     cout << x << endl;
36     return x; //now we need to return the good positive integer
37 }
-- INSERT -- 37,2 Bot
```

Global Variables

Do NOT use them!!!

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
1 #include <iostream>
2 #include <string> //c++ strings
3 #include <cstdlib> //atoi()
4
5 using namespace std;
6
7 //everyone can see this global variable, so you don't need local x or return
8 int x; //We should not make global variable for this!!!
9 void get_positive_int(); //function declaration/prototype
10
11 int main() {
12     get_positive_int(); //function call
13     cout << x << endl; //access the global variable, don't need local
14     return 0;
15 }
16 //function definition
17 void get_positive_int(){
18     string s; //create a string object
19     bool bad; //create a flag to indicate bad or good data
20
-- INSERT --
7,77 Top
```

Passing Arguments/Parameters

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
#include <iostream>
#include <string> //c++ strings
#include <cstdlib> //atoi()
4
using namespace std;
6
7 bool is_positive_int(string); //this function receives a string argument
8
9 int main() {
10     int x;
11     string s; //create a string object
12
13     do {
14         cout << "enter int: ";
15         cin >> s; //read data as a string, so it will never fail.
16     } while(is_positive_int(s)==false); //while it is not a good positive int
17
18     x=atoi(s.c_str());
19     cout << x << endl;
20
21     return 0;
22 }
6,0-1 Top
```

Passing Arguments/Parameters

```
2. ENGR
Re-attach Fullscreen Stay on top Duplicate
12
13     do {
14         cout << "enter int: ";
15         cin >> s; //read data as a string, so it will never fail.
16     } while(is_positive_int(s)==false); //while it is not a good positive int
17
18     x=atoi(s.c_str());
19     cout << x << endl;
20
21     return 0;
22 }
23
24 //This function does one thing now to check if a string is a positive int
25 bool is_positive_int(string s) {
26     //check that all the characters are 0-9 for a postivie int
27     for(int i=0; i<s.length(); i++)
28         if(!(s.at(i)>='0' && s.at(i)<='9')) {
29             return false; //if a char in the string is not 0-9, not positive
30         }
31     return true; //true it is a good positive int
32 }
-- INSERT --
31,53 Bot
```

Back to **break**, **exit**, and **return**

- **break** – used with switch and loops, breaking out of the closest associated case or loop(for, while, or do while). **This statement can only occur in a loop or case**, otherwise the compiler yells!
- **return** – leave the current function, which exits the program when in the main() function. You can put this **anywhere inside any function**, otherwise the compiler yells!
- **exit()** – exit the entire program, no matter where this is encountered. You can put this **anywhere inside any function**, as long as you include **<cstdlib>**, otherwise the compiler yells!