PFI lecture Function

- What is a function in programming language?
- From the user of a function perspective:
  - We outsource the work to be done to a function by calling the function. When the work is done, the function returns.
  - We use parameters or variables to supply information to a function.
  - A function may or may not return a value. A function may also return information via parameters.
- In fact, cin, cout, and getline that we have used are functions! The main program is a function declaration called from the operating systems.

PFI lecture ctrsing functions

- Need to include <cstring>, often this file is included in <iostream> so we may omit it.
- Examples of cstring library functions:
  - **strlen** is the name of the function for figuring out the number of characters in a cstring. It takes one cstring and returns the number of characters in the string excluding the terminating null character.

    char greeting[20] = “Hello, World!!”;

    int n = strlen(greeting); // n has value of 14
- Online or internet resource for documentation and example. www.cplusplus.com

PFI lecture ctrsing functions

- Examples of cstring library functions:
  - **strcpy** is the name of the function for copying cstrings. It takes two cstrings (left and right of an assignment): first cstring and second cstring such that the value of the first cstring takes the value of the second cstring. It returns the first cstring.

    char str[20] = {'H', 'i', '\0'};

    char greeting[20] = "Hello, World!!";

    strcpy(greeting, str);

    cout << greeting; // output is Hi

PFI lecture cstring functions

- Examples of cstring library functions:
  - **strcmp** is the name of the function for comparing two strings.
  - **strcat** is the name of the function for appending one string to another string
  - **strncpy** is the name of the function that is similar to strcpy.
  - **strstr** is the name of the function for searching a substring within a string.
- Use the online reference to learn the above.

PFI lecture math functions

- Need to include <cmath> in order to use math functions.
- Examples of math library functions:
  - **sqrt** is the name of the function for figuring out the square root of a value. It takes one double value and returns the square root of the value (type double).

    double std = sqrt(variance); // assume variance available

    std = sqrt(2); // std has value 1.41421
- Online or internet resource for documentation and example.

PFI lecture math functions

- Examples of math library functions:
  - **pow** is the name of the function for computing x^y.
  - **cos** is the name of the function for computing the cosine of an angle in radians.
  - **log** is the name of the function for computing the natural log (base e).
  - **Most math functions simply return results.**
- Use the online reference to learn the above.

PFI lecture Function: Users Perspective

- From the user of a function perspective:
  - Know the name of a function
  - Know what a function does, mainly, the relationship between the input to a function and the output from a function
  - Know how to pass information to a function, such as the number of arguments and their types, the order of the parameters.
  - Know how to get the information returned from a function, such as the return type, parameters their values might change.
  - Know the call of a function may be an expression
  - Know a function returns to the statement that invokes the function, then the next statement after the call is executed.

PFI lecture Function: Designers Perspective

- From designers of a function perspective:
  - Identify a common problem to be solved by a function
  - Figuring out the algorithm to efficiently solve the problem
  - Choose the name of a function
  - Choose how to pass information to a function, such as the number of arguments and their types, the order of the parameters.
  - Choose how to pass the information back from a function, such as using return or parameters and their types
  - Know the syntax in C++ of declaring a function
  - Know the semantics of parameter passing and function return

PFI lecture Function: Programmers Perspective

- As programmers, we are the users of “built-in” functions (designed by very experienced designers) provided by the systems, and at the same time we are designers of functions (user designed functions) for and used mainly by our own programs.
- Functions make our program modular and enhance understanding
- Functions provide facilities for abstraction, leading to code reuse and parameterization

Modularity example

- Hw3 with functions (parameters details omitted):

```
int main(){
      // declare variables
      get_course(course info);
      process_user_request(course info);
      return 0;
}
```

Basic syntax

- Function declaration:

```
return_type name(parameter list){
        // function body
}
```

- Example: a function that decides if a given integer is even or not.

```
bool even(int n){  // function declaration
        if (n%2 == 0)
                return true;
        else
                return false;
}
int main() {
        int x;
        if ( even(x) ) // calling function as an expression
                cout << x << " is even.";
}
```

Examples

- Note that the even function in the previous example follows a similar pattern as some of the math functions in the standard library. We could write a prime function that return true of the argument is a prime as follows.

```
bool prime(int n){
        // complete the rest as an exercise
}
```

- Example: a function that outputs a message and returns nothing.

```
void msg(string s){
        cout << “The value of the variable ” << s << “is:”;
}
int main() {
        int x = 10;
        string str = “Hey”;
        char c = ‘A’;
        msg(“x”);
        cout << x << endl;
        msg(“str”);
        cout << str << endl;
        msg(“c”);
        cout << c << endl;
}
```

Examples

- In debugging, we often need to check to see the values of our variables. The msg function in the previous example solves part of the problem. This example expands that further.

```
void check_int(string s, int m){
        cout << “The value of the variable ” << s << “is:” << m <<
   endl;
}
int main() {
        int x = 10;
        int y;
        check_int(“x”, x);
        y = x*x;
        check_int(“y”, y);
        }
```

- We may define similar functions for the remaining four types. For example,

```
void check_string(string s, string m){
        cout << “The value of the variable ” << s << “is:” << m <<
   endl;
}
```