Current Assignments

- Homework 2 is available and is due tomorrow (June 19th).

  Boolean expressions, if statements and the while loop.

- Project 1 due in 5 days (June 23rd)

  Write a binomial root solver using the quadratic equation.

This Time

- Homework 2 Examples
- Break, Continue
- Switch
- Control Structure Examples
- 30 minute Lab
- Functions
- Calling functions
- Anatomy of a function

Homework 2, examples

- Boolean Expressions

  if x is true, and y is false

  is this true or false?

x || y && !y

true || (false && !false) is true

Homework 2, examples

- Boolean Expressions
  if x is true, and y is false

x && y && !y

false

In fact this is false for all values of x and y
Boolean expressions that are **always** false are called “contradictions.”

Homework 2, examples

- Boolean Expressions
  if x is true, and y is false

x || y || !y

true

In fact this is true for all values of x and y
Boolean expressions that are **always** true are called “tautologies.”

Homework 2, examples

- Boolean Expressions
  if x is 9, and y is 3

$!(x < y)$

true

Could be rewritten as $(x >= y)$

Homework 2, examples

- Boolean Expressions
  for what values of bar and foo is this true?

((foo && !bar) || (bar && !foo)) && foo;

One way to solve this is with a truth table (Chapter 2, Page 125 of Deitel and Deitel)

((foo && !bar) || (bar && !foo)) && foo;

foo	**bar**	**((foo && !bar) || (bar && !foo)) && foo**
true	true	((true && !true) || (true && !true)) && true (false || false) && true is **false**
true	false	((true&& !false)||(false && !true)) && true (true || false) && true is **true**
false	true	((false&& !true)||(true && !false))&& false (false || true) && false is **false**
false	false	((false&& !false)||(false&&!false))&&false (false || false) && false is **false**

The expression is true only when foo is true and bar is false

(foo||!bar) && (foo||bar) == (foo || (!bar&&bar))

We can use a truth table

foo	**bar**	**(foo||!bar) && (foo||bar) == (foo || (!bar&&bar))**
T	T	(T || !T) && (T || T) == (T || (!T && T)) (T && T) == (T || F) is **true**
T	F	(T || !F) && (T || F) == (T || (!F && F)) (T && T) == (T || F) is **true**
F	T	(F||!T) && (F||T) == (F || (!T&&T)) (F && T) == (F || F) is **true**
F	F	(F||!F) && (F||F) == (F || (!F&&F)) (T && F) == (F || F) is **true**

This is tautology. It is always true.

Break

- The break command tells the program to “break” out of the current control structure
- Use break sparingly because it disrupts the regular flow of control and can lead to spaghetti code.
- Typically used to quit a loop early because of some special circumstance not handled by the loops guard condition.

Break, example

```
for (int i = 0; i < 10; i++)
{
   if ( i == 3 )
   {
        break;
   }
}
```

Continue

- The “continue” statement when executed skips to the end of a control structure but does not exit the control structure.
- Continue is not commonly used.

Continue, example

```
for (int i = 0; i < 10; i++)
{
   cout << x;
   if ( i > 5)
   {
        continue;
   }
   cout << y;
}
```

The switch statement

- The switch statement is a holdover from C
- “Switch” can be used instead of “if … else” as a selection control structure
- Can only be used when the selection condition is that some variable is equal to a whole number

Switch, example 1

```
int choice = 1;
switch( choice )
{
   case 0: cout << “choice was 0” << endl;
        break;
   case 1: cout << “choice was 1” << endl;
        break;
   case 2: cout << “choice was 2” << endl;
        break;
   default: cout << “unknown choice” << endl;
}
```

Switch, example 2

```
int choice = 1;
switch( choice )
{
   case 0: cout << “choice was 0” << endl;
        break;
   case 1: cout << “choice was 1” << endl;

   case 2: cout << “choice was 2” << endl;
        break;
   default: cout << “unknown choice” << endl;
}
```

Switch, example 3

```
int choice = 1;
switch( choice )
{
   case 0: cout << “choice was 0” << endl;

   case 1: cout << “choice was 1” << endl;

   case 2: cout << “choice was 2” << endl;

   default: cout << “unknown choice” << endl;
}
```

Switch, example 4

```
char choice = 't' ;
switch( choice )
{
   case 'a' : cout << "choice was a" << endl;
        break;
   case 'b' : cout << "choice was b" << endl;
        break;
   case 't' : cout << "choice was t" << endl;
        break;
   default: cout << "unknown choice" << endl;
}
```

Switch, example 5

```
int choice = 4;
switch( choice )
{
   case 1: cout << “choice was 1” << endl;
        break;
   case 2: cout << “choice was 2” << endl;
        break;
   case 3 || 4: cout << “choice was 3 or 4” << endl;
        break;
   default: cout << “unknown choice” << endl;
}
```

Switch, example 6

```
int choice = 4;
switch( choice )
{
    case 1: cout << “choice was 1” << endl;
        break;
    case 2: cout << “choice was 2” << endl;
        break;
    case 3:
    case 4: cout << “choice was 3 or 4” << endl;
        break;
    default: cout << “unknown choice” << endl;
}
```

Lab

- Write the program "maxvalue"
- Finds the maximum of a group of numbers entered by the user (don't limit the number of values you program can read).

Example run: (user input is in **bold**)

Enter number: **8**

Enter another number (y/n)? **y**

Enter number: **7**

Enter another number (y/n)? **n**

The largest number you entered was 8

Use a "do…while" as you main loop.