Selection (if/else/switch) and booleans

- most programs need some form of decision making so they can examine current conditions and choose what to do next
- decisions generally hinge on a true/false condition check (if X is true then do Y, otherwise do Z)
- two forms of decision control in C++: if/else statements and switch statements
- Boolean logic is a system of true/false logic, using operators for logical and, or, and not (e.g. if X is true and Y is true then do A)
- Boolean variables will be introduced to hold true/false values

Two-way control: if/else

- the most common decision making structure tests one true/false condition: if it is true then do one set of actions, otherwise do the other

```
if (x < y) {
   cout << x << " is smaller than " << y << endl;
} else {
   cout << y << " is smaller than " << x << endl;
}
```

- exactly one of the two blocks of code is executed
- each of the two blocks can have multiple statements inside { }

Multi-way control: adding else if

- sometimes we want to check multiple conditions

```
if (x == y) {
    cout << x << " is equal to " << y << endl;
} else if (x < y) {
    cout << x << " is smaller than " << y << endl;
} else {
    cout << y << " is smaller than " << x << endl;
}
```

- we can have as many “else if” combos as we like
- the code block for the first true condition executes (the else block runs if no condition is true)

Example: checking bounds

```
// get the user to enter a value in a specific range,
// check that it is actually within range

cout << “enter a number between 1 and 10” << endl;
float x;
cin >> x;
if (x < 1) {
   cout << x << “ is too small (less than 1)” << endl;
} else if (x > 10) {
   cout << x << “ is too big (larger than 10)” << endl;
} else {
   cout << x << “ is a valid entry” << endl;
}
```

Example: min, max

```
// return the smaller of the two passed parameters
int min(int a, int b)
{
   if (a < b) {
      return a;
   } else {
      return b;
}

// return the larger of two passed parameters
int max(int a, int b)
{
   if (a < b) {
      return b;
   } else {
     return a;
   }
}
```

Example: sorting 3 params

```
// sort params in increasing order
void sort3(float &x, float &y, float &z)
{
   float small, large, middle;
   if (x < y) {
      small = x;
      large = y
   } else {
      small = y;
      large = x;
   }
   //
   if (z < small) {
      middle = small;
      small = z;
   } else if (large < z) {
      // continues on right...
                                    // continues from left ...
                                     middle = large;
                                     large = z;
                                   } else {
                                     middle = z;
                                   }
                                   x = small
                                   y = middle;
                                   z = large;
                                }
```

Compound logic expressions

- we can group logical conditions together with logical and (the && symbol) or logical or (the || symbol)
- to test if x is less than y AND y is also less than z:

```
if ((x < y) && (y < z)) {
```

- to test if a is less than b OR a is less than c:

```
if ((a < b) || (a < c)) {
```

- to take the opposite of a condition, e.g. if it is NOT the case that a < b

```
if (! (a < b) ) {
```

Common comparison operators

- less than (a < b)
- greater than (a > b)
- less than or equal to (a <= b)
- greater than or equal to (a >= b)
- equal to (a == b)
- not equal to (a != b)

Multiway checks for values

- If we have a variable that might have one of a specific set of values we could check with a series of if/else's, e.g.

```
if (x == 10) {
   // code for case 10
} else if (x == 15) {
   // code for case 15
} else if (x = 17) {
   // code for case 17
} else {
   // code for any other value
}
```

switch statements

- an alternative to the previous use of else/ifs
- we use a “switch” on the variable in question, listing cases

```
switch (x) {
   case 10:  // code for case 10
        break;
   case 15:  // code for case 15
        break;
   case 17:  // code for case 17
        break;
   default:  // code for all other cases
        break;
}
```

break in switch statement

- break is used to indicate the end of each case, otherwise it goes on and runs the code for the next case too
- can be used to group values together if the have same behaviour, e.g. suppose command is a char variable and we don't care if the user enters in upper or lowercase:

```
switch (command) {
    case 'q':
    case 'Q':  // code for the Q or q commands
         break;
    ...etc....
}
```

Boolean variables

- sometimes it is handy to store the condition check in a variable (so we can remember it as the basis for some future decision)
- we use variables of type *bool*, and can assign values of *true* or *false*

```
// suppose data is supposed to be between min and max
bool isDataOK = true;
if ((data < min) || (data > max)) {
   isDataOK = false;
}
// lets us remember for later whether or not the data was ok
// anytime later on we could check it using simply
if (isDataOK) {
```