Discussion 3

SAKIB

const variables

Constants are "variables" (in quotes because constants don't vary) that, once declared, cannot change value anywhere in the scope.

They are declared as:

const <type> <name> = <value>;

Why const?

The One Change, One Place philosophy whereby we need only change one variable when we want to replace a value throughout our code. Forcing constant variables to be unable to change protects us from committing errors by changing values that weren't meant to be changed

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   const int X = 6;
   const double PI = 3.1415;
   const string STUFF = “STRSTRSTR";
}
```

```
// Will the following code compile?
// If so, what will it print?

#include <iostream>
using namespace std;

int main () {
  const int DER = 6;
  DER = 5;
}
```

Returning to conditionals

Remember from lecture

Incorrect ways to use conditionals

```
if (country == "US"  ||  "Canada")      // error! ☹
if (country == "US"  ||  == "Canada")   // error! ☹
if (age >= 18  &&  < 21)                // error! ☹
```

If you check a variable for multiple conditionals, you must do it one at a time

```
if (country == "US"  ||  country=="Canada")     //  yay! ☺
if (age >= 18  && age < 21)                 // yay! ☺
```

Unless you group conditionals (using parentheses), your code will evaluate conditionals in this order:

Logical Binary Operators (e.g. <, ==, etc.)

Logical AND (&&)

Logical OR (||)

Reduce conditionals for clear code

Don’t repeat your conditional

If you have some similar component between two or more conditions in an if-ladder, you might be able to condense them into a single parent condition. Is

```
if (i > 5 && j < 10) {
    // Action A
}
else if (i > 5 && j > 15) {
    // Action B
}
```


```
if (i > 5) {
    // Everything inside here implies that
    // i is greater than 5
    if (j < 10) {
        // Action A
    }
    else if (j > 15) {
        // Action B
    }
}
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int x = 1, y = 2, z = 3;

    // NB: You'll usually want to use grouping
    // to break apart a conditional of this
    // complexity--even just for clarity's
    // sake
    if (x == y || x < 3 && z < y || z != y) {
        cout << "You're in here!" << endl;
    }
    else {
        cout << "You're in there!" << endl;
    }
}
```

Boolean types

What we as humans know as "true" and "false" can be represented in a variety of C++ expressions.

Type bool, short for Boolean, are variables that hold only the value true or the value false. These are still represented as 1 and 0 respectively

Assigning any number other than zero converts to 1 (True)

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   bool iLikeNums = 50 * 20;

   cout << iLikeNums << endl;
}
```

Because 0 and 1 are the main truth and false values, we can store comparison results in variables

```
#include <iostream>
using namespace std;

int main() {
    int conditionOutcome,
    e = 2,
    n = 4;

    // Comparisons can be stored
    // as ints
    conditionOutcome = e > n;
    cout << conditionOutcome << endl;
    conditionOutcome = e < n;
    cout << conditionOutcome << endl;
}
```

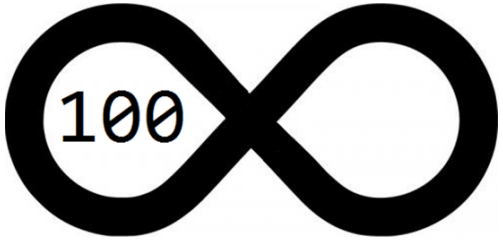
Loops

Repeat code

Continue as long as **looping condition** is **true**

3 types of loops!!!!

- while
- do while
- for
- Almost all the same



While loop

while([**running condition**]) {

    [code]

}

The while loop executes the code block as long as the running condition is true.

- At every iteration check condition first then execute

Do while loop

Always executes **once**!!!!!!!!!!!!!!!

```
do {
        [code]
} while( [running condition]) ;
```

Exectute the code block **first** then check the condition

For loop

```
for(initialization; stay-in-loop-condition; prepare-for-next-iteration )
{
        Statements
}
```

Just like a while loop, except we initialize the iterator, define the conditional, and define the post-iteration behavior all in the signature

What gets printed?

```
int i = 4;

while (i < 4) {
    cout << "i am in the while loop" << endl;
}

do {
    cout << "i am in the do-while loop" << endl;
} while (i < 4);
```

When you are using loops, it is a good idea to use an **iterator** to keep track of how many times you want to execute a statement

```
#include <iostream>
using namespace std;

int main() {
    int i = 1;

    // print 1 to 9 each on a separate line
    while (i != 10) {
        cout << i << endl;
        i++;
    }
}
```

The for loop equivalent of the previous example

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i != 10; i++){
    cout << i << endl;
    }
}
```

Practice

Use a loop to multiply an integer

4*3 = 4 + 4 + 4

Raise an integer to a power of n?

On Characters & Strings

Characters are single letters or symbols like 'a', or '|', and include special characters like the new line character '\n'. We use **single quotes** to designate characters.

Strings consist of some n characters that are "strung" together, where n >= 0. We use **double quotes** to designate **strings**

Characters and strings

Strings allow you to use the [] operator

You can extract characters from strings

Or change them!

Important note!! In strings, the index of the first character starts with 0 not 1!!

Zero the hero

First the worst

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string me = "Sakib = Cool!";

    cout << me[0] << endl;
    cout << me[4] << endl;

    me[8] = 'F';
    cout << me << endl;
}
```

A useful library. `#include <cctype>`

There are a variety of operations we can use with characters in the cctype library.

isalpha asks if this character is a letter.

isdigit asks if this character is a number.

isalnum asks if this character is alphanumeric. (is it a number or a letter?)

isupper asks if this character is an uppercase letter.

islower asks if this character is a lowercase letter.

```cpp
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

int main() {
    string stringy = "Hi5";

    if (isalpha(stringy[1]))
    cout << "Its an alpha" << endl;

    if (isupper(stringy[0]))
    cout << "Its upper case" << endl;
}
```

Types of strings

C Strings are arrays of characters **that possess a terminating 0 byte** designating the end of the string.

- Don’t worry about these…… yet MWAHAHA

C++ Strings are **objects** representing arrays of characters, and do NOT possess the terminating 0 byte

- You can freak out now

Big X

Design a program that writes a large X made of little Xs on a 5 by 5 grid

http://ideone.com/ZDGfVZ

Examples

Design a program that asks the user for text input and then swaps all capital letters for their lowercase equivalents, and vice-versa

http://ideone.com/NpwVWj

Guessing Game

Try to guess the number the computer is thinking of, we generate a number between 0 and MAX_CHOICES. We ask users to guess and tell them if they are close or way off. Players have MAX_TRIES to guess the number

Start with this template http://ideone.com/SQucdQ

We'll offer hints if a user's guess is 1 away from the number or way off (difference is 2 or more)

If the user wins, we congratulate them and end the program there.

Solution: http://ideone.com/TnqFWr