CS 31 Discussion

ABDULLAH-AL-ZUBAER IMRAN

WEEK 2: STRINGS AND CONTROL FLOW

Recap

Introduction: computer program, programming language

First C++ program

Variables and value assignments

Identifiers and naming rules

Comments

Errors: compile errors and logic errors

Discussion Objectives

Review and practice things covered during lectures

- Strings, if statements, conditional loops
- Coding examples

Clarifications for project assignments (Project 2)

Time for you to ask questions!

Good Coding Practices

- Have descriptive variable names
- Use indentation to clarify meaning
- Have short, descriptive comments
- Incremental development

Incremental development tips

- Write small blocks of code at a time and test them with multiple types of input
- If you have multiple errors, test them sequentially and recompile as you fix mistakes. Sometimes fixing one mistake will also fix later ones.
- Save often!

Input/Output

```
// i/o example
#include <iostream>
using namespace std;

int main ()
   {
      int i;
      cout << "Please enter an integer value: ";
      cin >> i;
      cout << "The value you entered is " << i<<endl;
      cout << " and its square is " << i*i << endl;
    }
```

```
Please enter an integer value: 7
The value you entered is 7
And its square is 49
```

Strings

Strings are objects that represent a sequence of characters.

```
#include <string>
```

Examples:

- “Jie”
- “”
- “ ”
- “\n”

A few things you can do with strings.

Operation	What it does	Example
`string s = “hello”;` `string s2 = “!!!”;`	Declare strings s and s2	
`s.length()` or `s.size()`	Return the length of s	`cout << s.size(); // prints 5`

`s[i]` or `s.at[i]`	Return i-th character. (`i` should be integer between 0 and size-1 (inclusive))	`cout << s[1]; // prints ‘e’`
		`cout << s.at(0); // prints ‘h’`

`s + s2`	Concatenate two strings	`cout << s + s2; // prints “hello!!!”`

Strings (Cont'd)

Question: Will this program compile? If so, what's the output?

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

int main(){
  string name;
  getline(cin, name);
  cout << "Hello! " << name << endl;
}
```

```
Input: Jay
Output: Hello! Jay
```

Compiler is being nice to you. It detects that you are using strings and includes the library `<string>` for you. It is always the best practice to include `<string>` when you are using strings. (`<iostream>` also includes part of definitions in `<string>`)

Strings (Cont'd)

Question: Will this program compile? If so, what's the output?

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

int main (){
  string text = "hi",
         blank = "",
         space = " ",
         newLine = "\n",
         result;
  result = text + blank + space + "!" + newLine;
  cout << result << endl;
  cout << "---" << endl;
}
```

Output:

```
hi !


---
```

Strings (Cont'd)

Question: Will this program compile? If so, what's the output?

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

int main () {
  string test = "hi";
  int five = 5;
  test = test + five;
  cout << test << endl;
}
```

Output:

This program won't compile.
The operands for concatenation operator '+' should be two strings.

Testing empty string

Strings can be tested to see if they're empty

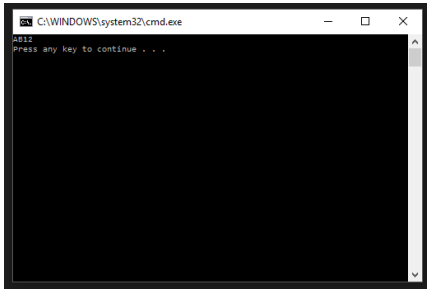
```
int main(){

cout << "What is your name? ";


string name;
getline(cin, name);


 if (name == "")
        cout << "You didn't type a name!" << endl;
 else
        cout << "Hello, " << name << endl;

 }
```

Stream

In C++, I/O performed by using streams. A **stream** is a “stream of data” in which character sequences are “flow into” or “flow out off”.

`cout` is the standard output stream which by default accesses the screen.

`cin` is the standard input stream which by default accesses the keyboard.



‘A’ ‘B’ ‘1’ ‘2’ …

cout

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

int main() {
  cout << "AB12" << endl;
}
```

… ‘2’ ‘1’ ‘B’ ‘A’

cin

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

int main() {
  string c;
  cin >> c;
}
```

Stream

`cin >> var;` command accesses input characters, ignores whitespace, and ignores the newline at the end of the user's input. We use this to get numerical input, and store it in variable "`var`".

`getline(cin, s);` command consumes all characters up to, and including, the newline character. It then throws away the newline, and stores the resulting string in `s`. We use this to gather string inputs. (requires `<string>` library)

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

int main () {
  string inputString;
  int inputInt;
  cout << "Enter a number: ";
  cin >> inputInt;
  cout << "Input was: " << inputInt << endl;
  cout << "Enter a string: ";
  getline(cin, inputString);
  cout << "Input was: " << inputString << endl;
}
```

```
Input:
32
world
Output:
Enter a number: 32
Input was: 32
Enter a string: Input was:
```

Stream

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main () {
6   string inputString;
7   int inputInt;
8   cout << "Enter a number: ";
9   cin >> inputInt;
10  cout << "Input was: " << inputInt << endl;
11  cout << "Enter a string: ";
12  getline(cin, inputString);
13  cout << "Input was: " << inputString << endl;
14}
```

```
Input:
32
world
Output:
Enter a number: 32
Input was: 32
Enter a string:Input was:
```

At line 9:

The value 32 is stored in the variable `inputInput`, '\n' is left in the `cin` stream.

At line 12:

The `getline(cin, inputString)` consumes '\n', then discards the newline '\n', and stores the string left to `inputString`. Since there is nothing left, null character '' is stored to `inputString`.

Use `cin.ignore(n,pattern)` when we've used `cin` and then directly after use `getline`.

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main () {
6   string inputString;
7   int inputInt;
8   cout << "Enter a number: ";
9   cin >> inputInt;
10  cout << "Input was: " << inputInt << endl;
11  cin.ignore(10000, '\n');
11  cout << "Enter a string: ";
12  getline(cin, inputString);
13  cout << "Input was: " << inputString << endl;
14}
```

```
Input:
32
world
Output:
Enter a number: 32
Input was: 32
Enter a string: world
Input was: world
```

To fix it, we will need to consume the extra newline.

`cin.ignore(n, pattern)` ignores n characters or until the first encountered instance of pattern from input stream.

At line 11:

`cin` `'\n'` `cin`

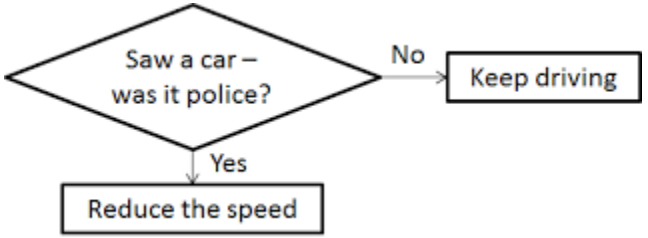
`cin.ignore(10000, '\n');` command consumes either 10000 characters, or discards all the characters until the first encountered `'\n'` (inclusive).

At line 12:

`cin` `'w' 'o' 'r' 'l' 'd' '\n'` `inputString` `world`

The `getline(cin, inputString)` consumes "world\n", then discards the newline `'\n'`, and stores the string left to `inputString`.

If-else statements



In many cases, you want your program to behave differently based on some condition. `if-else` statements are what you would use in those cases. If takes the following form:

```
if (condition)
  statement;
```

```
if (condition){
  statement1;
  statement2;
  …
}
```

Use curly brackets when there are multiple statements in the if-block

```
if (condition1){
// Execute if condition1 is evaluated as true
  statement1;
  statement2;
  …
} else {
// Execute if condition1 is evaluated as false
  statementn;
}
```

```
if (conditional1){
// Execute if conditional1 is evaluated as true
} else if (conditional2){
// Execute if conditional1 is evaluated as false AND
// conditional2 is evaluated as true
} else {
// Execute if conditional1 is evaluated as false AND
// conditional2 is evaluated as false
}
```

If-else statements

Boolean statements

- The condition in the if statement must be a yes-or-no question. We express this question in the form of a **boolean expression**, which evaluates to either true or false.

symbol	Meaning
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to
&&	AND
||	OR

"Equal to" symbol consists of two '='s, not one.
=> And =< are invalid.

If-else statements

Question: Suppose that `x==5`, `y==6`, and `z==7`. Can you evaluate the following expressions?

Boolean statement	Answer
`x >= y`	False
`x == y`	False
`x - y > 10`	False
`x > y && y < z`	False
`((x != y) || (x > y)) && (y == z)`	False
`(x != y) || ((x > y) && (y == z))`	True

If-else statements

Question: Will this program compile? If so, what's the output?

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

int main () {
  int age;
  cin >> age;
  if (age = 30){
    cout << "Your age: 30" << endl;
  } else {
    cout << "You are not 30" << endl;
  }
}
```

```
Input: 40
Output:
Your age: 30
```

The result of assignment `age = 30` is the value of the age after the assignment.
30 is a non-zero number, the result of the condition is therefore `true`. If the value assigned is 0, the condition is `false`.

Loops

Loops let you repeat the same or similar task multiple times. Three primitive loops in C++: `while`, `do-while`, and `for`.

`while` loop

```
while (condition)
  body;
```

1. Evaluate the `condition`.
   If `true`,
   2. run the body.
   Go back to 1,
   If `false`,
   exit the while loop.

`do-while` loop

```
do {
  body;
} while (condition);
```

1. Execute the body.
2. Evaluate the condition
   If `true`,
   Go back to 1,
   If `false`,
   exit the while loop.

Don't forget the ';' here.

Notice: The body in do-while loop will be executed once no matter what.

Loops

Question: Will this program compile? If so, what's the output?

Loops

Question: Will this program compile? If so, what's the output?

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

int main() {
  int x = 0, y = 10;
  while (x < y) {
    x++; // equivalent to x = x + 1
    cout << x << " ";
  }
  cout << endl;
  return 0;
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10
```

Loops

`for` loop

```
for (initialization; condition; update)
  body;
```

1. Execute `initialization`.
2. Evaluate the `condition`.
   If `true`,
   3. Run the body.
   4. Do the `update`.
   Go back to 2.
   If `false`,
   exit the `for` loop.

Loops

Question: Will this program compile? If so, what's the output?

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

int main () {
  int iterations = 10;
  for (int i = 0; i < iterations; i++) {
    if (i % 2 == 0) {
      cout << i << endl;
    }
  }
}
```

```
Output:
0
2
4
6
8
```

%: modulus operator -> compute the remainder that results from preforming integer division

Loops

`for`-to-`while` conversion

- If you can do something using a `while` loop, you should be able to write a `for` loop equivalent, and vice versa.

```
for (initialization; condition; update) {
  body;
}
```

```
initialization;
while (condition) {
  body;
  update;
}
```

Loops

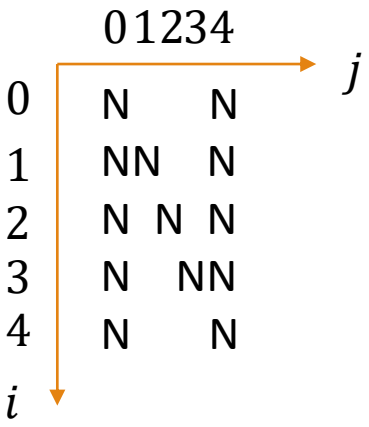
Question: Convert the following `for loop` into a `while` loop.

```
for (int i = 0; i < 50; i += 2) {
  cout << "Hello" << endl;
}
```

```
int i = 0;
while(i < 50){
  cout << "Hello" << endl;
  i += 2;
}
```

Programming challenge

Design a program that writes a 5 by 5 capital letter N made up of the character ‘N’. Output should be:



Hints:

$$\begin{cases} j = ?\,, print\ 'N' \\ j = ?\,, print\ 'N' \\ j = ?\,, print\ 'N' \end{cases} \quad 0 \le i < 5$$

Expressions

An expression is a combination of variables, constants, operators, and function call.

```
int z = x + y        //arithmetic

A > B                // relational

a == b              //logical
```

Operators

Relational: Operators are used to compare the values of two variables or expressions depending on their relations.

a >= b	(greater than or equal)
A + B < C	(less than)
x == y	(equal to)

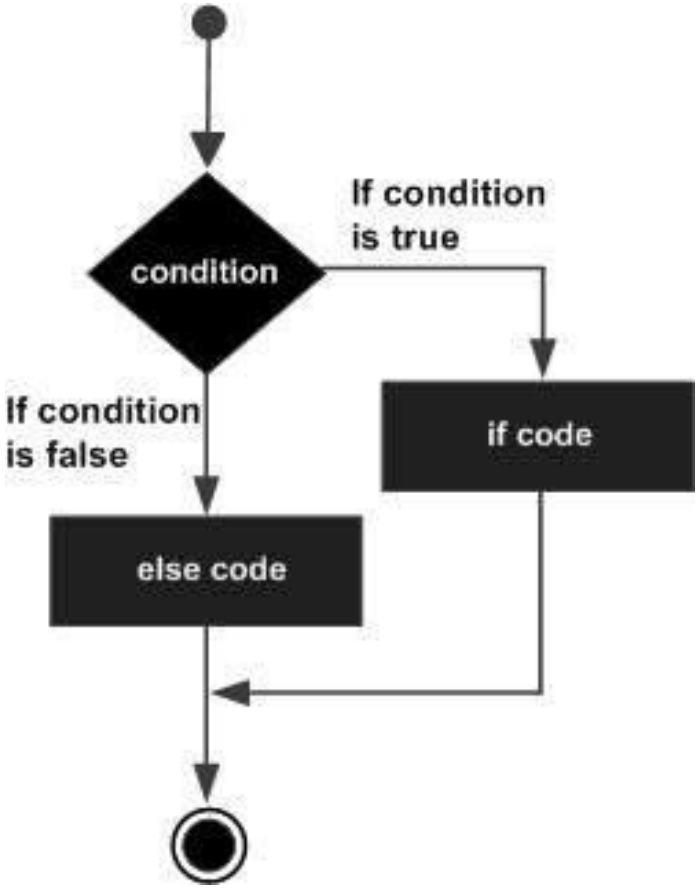
Logical: Operators used with one or more operands return either value 0 (False) or 1 (True)

&&	(AND)
||	(OR)
!	(NOT)

if statement syntax

```
if (condition)
        statement_to_execute_if_condition_is_true;
 or

 if (condition)
        { statement 1; statement 2; _ _ _ _; }
```

If/else syntax




```
if (condition)
        statement; // condition is true
else
        statement;  // condition is false
```

if..else example

```
//Checking even or odd number
#include<iostream.h>
int main () {
  int num;
  cout<<“Enter a number:”<<endl;
  cin>>num;

  if(num % 2 == 0)
       cout<<“Even number”<<endl;
  else
       cout<<“Odd number”<<endl;
```

Nested if statement

```
#include<iostream.h>
int main () {
  int num;
  cout<<“Enter a number:”<<endl;
  cin>>num;
  if (num > 0) {
            if(num % 2 == 0)
                        cout<<“Even number”<<endl;
            else
                        cout<<“Odd number”<<endl;
                  }
  else
  {
            } //What would be the purpose of else here?
```

If..else..if Ladder

```
if (expression)
        statement_1
else if (expression)
        statement_2
        ..
        ..
else
        statement_3
```

Question: When do we need if..else..if ladder?

Example

```
if (x > 0)
      cout << "x is positive";
 else if (x < 0)
      cout << "x is negative";
else
      cout << "x is 0";
```

Review

A computer program is written by a computer programmer in a programming language.

- E.g.: C++, C, java, etc.

What is a programming language?

	Human Language	Programming Language	Machine Language
	English Spanish Italian ...	C C++ Java ...	binary numbers
for humans	easy	medium	difficult
for computers	difficult	medium	easy

It is a language of “medium difficulty” for both us and computers. We use it to represent the logic of a program.

Project 2

The zip file you submit must follow the instructions exactly. (Pay attention to the name of each cpp file and your zip file)

Late submissions will be penalized.

Your code should run successfully under two compilers: g++ with linux and either Visual C++ or clang++ (Xcode).

Demo of g++

Project 2

Get input in exact order/manner specified in instructions

Output “---” for all cases

No loops

If bad input, do not keep prompting

Only write error message for earliest erroneous input (do not write others)

Do not print the rental charge if the input is invalid

Thanks!