

CS 31 Discussion 1J

ABDULLAH-AL-ZUBAER IMRAN & RYAN YANG

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
- Practice problems

Clarifications for project assignments (Project 2 & hw2)

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

```
6 // i/o example
7
8 #include <iostream>
9
10 using namespace std;
11
12
13 int main ()
14 {
15     int i;
16     cout << "Please enter an integer value: ";
17     cin >> i;
18     cout << "The value you entered is " << i<<endl;
19     cout << " and its square is " << i*i << endl;
20 }
```

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
<code>string s = "hello"; string s2 = "!!!";</code>	Declare strings s and s2	
<code>s.length()</code> or <code>s.size()</code>	Return the length of s	<code>cout << s.size(); // prints 5</code>
<code>s[i]</code> or <code>s.at[i]</code>	Return i-th character. (i should be integer between 0 and size-1 (inclusive))	<code>cout << s[1]; // prints 'e' cout << s.at(0); // prints 'h'</code>
<code>s + s2</code>	Concatenate two strings	<code>cout << s + s2; // prints "hello!!!"</code>

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

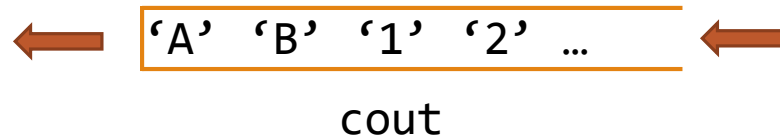
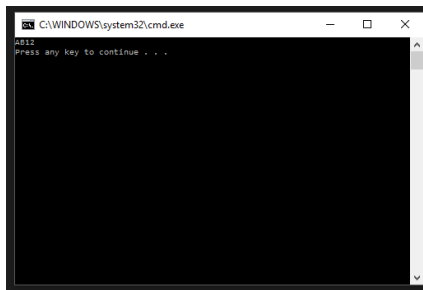
```
int main(){  
    cout << "What is your name? ";  
  
    string name;  
    getline(cin, name);  
  
    if (name == "")  
        cout << "You didn't type a name!" << endl;  
    else  
        cout << "Hello, " << name << endl;  
}
```

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”.

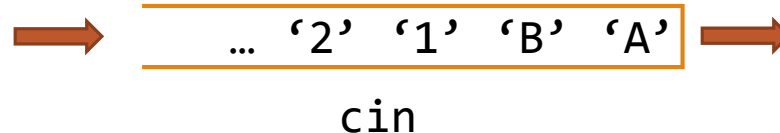
`cout` is the standard output stream which by default accesses the screen.

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



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

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



```
#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:

cin

inputInt

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

cin

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.

cin

inputString

Stream

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');
12    cout << "Enter a string: ";
13    getline(cin, inputString);
14    cout << "Input was: " << inputString << endl;
15}
```

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`

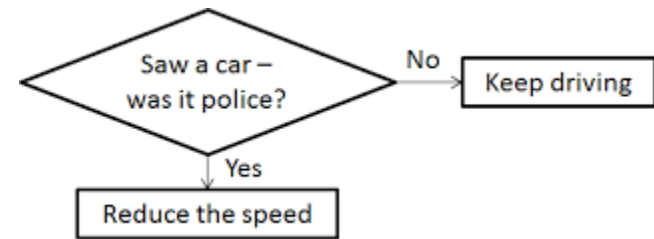
`cin.ignore(10000, '\n');` command consumes either 10000 characters, or discards all the characters until the first encountered `'\n'` (inclusive).

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;
```

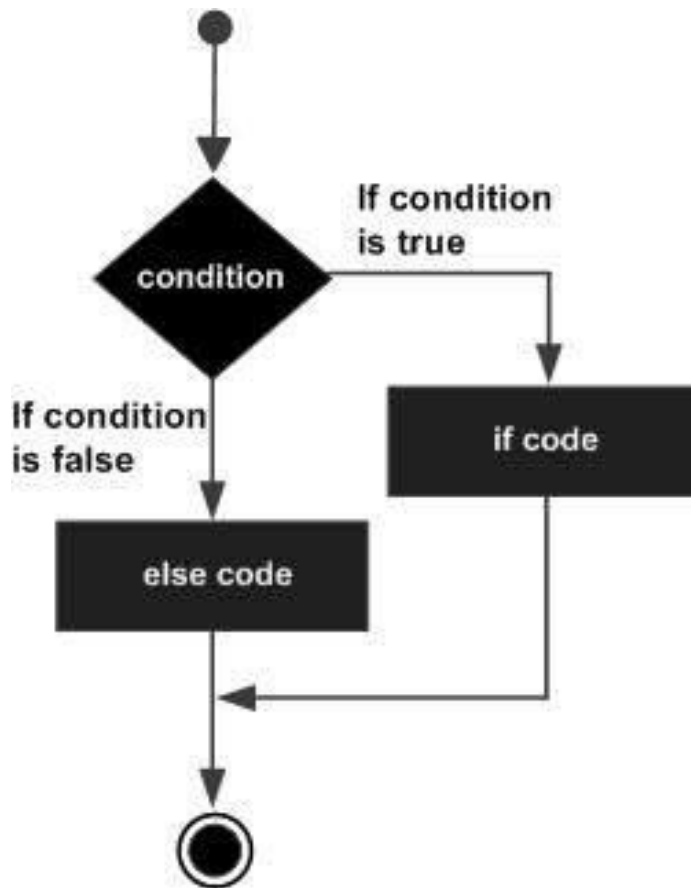
```
if (condition){
    statement1;
    statement2;
    ...
}
```

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 syntax



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

Boolean statements

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

Boolean statement	Answer
$x \geq y$	False
$x == y$	False
$x - y > 10$	False
$x > y \ \&\& \ y < z$	False
$((x \neq y) \ \ (x > y)) \ \&\& \ (y == z)$	False
$(x \neq y) \ \ ((x > y) \ \&\& \ (y == z))$	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.

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 Ladder

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

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

Example

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

Switch Statements

The `switch` statement is used to check for a value among a number of possible constant expressions.

The value of the expression you use should be an integer (`int`), or something that is equivalent to an integer (`char`).

It is something similar to concatenating if-else statements, but limited to constant expressions.

```
switch (expression) {  
    case constant1:  
        group-of-statements-1;  
        break;  
    case constant2:  
        group-of-statements-2;  
        break;  
    .  
    .  
    .  
    default:  
        default-group-of-statements;  
}
```

Switch example	If-else equivalent
<pre>switch (x) { case 1: cout << "x is 1"; break; case 2: cout << "x is 2"; break; default: cout << "value of x unknown"; }</pre>	<pre>if (x == 1) { cout << "x is 1"; } else if (x == 2) { cout << "x is 2"; } else { cout << "value of x unknown"; }</pre>

Switch Statements (Cont'd)

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

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

int main(){
    int x;
    cin >> x;
    switch (x) {
        case 1:
            cout << "x is 1";
            break;
        case 2:
        case 3:
            cout << "x is 2 or 3";
            break;
        default:
            cout << "x is not 1, 2 nor 3";
    }
}
```

Input: 1

Output:

x is 1

Input: 2

Output:

x is 2 or 3

Input: 3

Output:

x is 2 or 3

Input: 4

Output:

x is not 1, 2 nor 3

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)1  
    body;2
```

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

do-while loop

```
do {1  
    body;  
} while (condition);2
```

Don't forget the ';' here.

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

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;1 condition;2 update)4  
    body;3
```

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 performing 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;  
}
```


Project 2

The zip file you submit must follow the instructions **exactly**.
(Pay attention to the name of the 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).

Project 2

Get input in exact order/manner specified in instructions:

→gather the model value, the kind value, the color value, the wheels value, the leather value and an autopilot value, in that specific order.

If bad input, do not keep prompting, just write the required message

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

Do not print the purchase price if any of the inputs is invalid

Do not use any loops in this program

Homework2

Don't run the code, try to trace through and figure out by hand

Problem2: Don't write more than one brief English sentence. Remember English, nothing about the code Example: This program prints a picture of a circle

Problem 3,4,5: For conversion from one loop to another, make sure they are equivalent. That means you will get exactly the same output in either cases

Discussion Slides

You can find today's discussion slides at

https://github.com/zubaerimran/W20-CS31-1J/blob/master/week2/winter20_cs31_w2.pdf

Github repo for all the discussion materials:

<https://github.com/zubaerimran/W20-CS31-1J>

*Some of the materials presented have been taken from previous TA discussions

Thanks!

Questions?