Outline

- Problems from Project 1
- Review of lecture
  - String, char, stream
  - If-else statements
  - Switch statements
  - loops
- Programming challenge
Problems from Project 1

• Q1: Should I write `return 0;` at the end of the `main()` function?
• A: No need to write it in C++ program.

Operating System (Windows/Mac OS/Linux)

Main function will return integers to the operating system.
Zero – The program executes w/o errors
Non-zero – The program executes w/ errors.

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

int main(){
    cout << "Hello World!" << endl;
    return 0;
}
```

Operating System (Windows/Mac OS/Linux)

-1/0/1/2/...
Problems from Project 1

Still works.
The compiler will handle the case and add return 0; for you at the end of the main() function.

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

int main()
{
    cout << "Hello World" << endl;
    // return 0;
}
```

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

int main()
{
    int age;
    cin >> age;
    if (age < 0){
        cout << "Please input nonnegative number!" << endl;
        return 1;
    }
    cout << age << endl;
    return 0;
}
```
Problems from Project 1

• Q2: What are these lines in the console? (If you are using Visual Studio)

![Console Output]

• A: The text “Press any key to continue...” is added by IDE, and there is not need to worry about it confounding your answers.
String

- **Strings** are objects that represent a sequence of characters.
- `#include <string>`
- Examples:
  - “Jie”
  - “”
  - “ ”
  - “\n”
- A few things you can do with strings.

<table>
<thead>
<tr>
<th>Operation</th>
<th>What it does</th>
<th>Example</th>
</tr>
</thead>
</table>
| string s = “hello”;
string s = “!!!”;
| Declare strings s and s2 |                                                       | cout << s.size(); // prints 5 |
| s.length() or s.size() | Return the length of s                                | cout << s[1]; // prints ‘e’     |
| s[i] or s.at[i]     | Return i-th character. (i should be integer between 0 and size-1 (inclusive)) | cout << s.at(0); // prints ‘h’ |
| s + s2              | Concatenate two strings                                | cout << s + s2; // prints “hello!!!” |
String

• **Question:** Will this program compile? If so, what’s the output?

```cpp
#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>`)
• **Question:** Will this program compile? If so, what’s the output?

```cpp
#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 !
---
```
• **Question:** Will this program compile? If so, what’s the output?

```cpp
#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.
• **Characters** are single letters or symbols like ‘a’, or ‘|’, include special characters like the new line character ‘\n’. We use single quotes to designate characters.

• Characters can be represented by a unique integer value.

• ASCII (American Standard Code for Information Interchange) defines the mapping between characters and integers.

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

int main () {
    int x = '0';
    // Note that x is an integer and '0' is a character!
    cout << x << endl;
}
```

Output:

48
Here are some functions you can call on characters, after including `<cctype>` library.

<table>
<thead>
<tr>
<th>Operation</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>char c;</td>
<td>Declare a character c</td>
</tr>
<tr>
<td>isspace(c)</td>
<td>True if c is a whitespace character</td>
</tr>
<tr>
<td>isalpha(c)</td>
<td>True if c is a letter</td>
</tr>
<tr>
<td>isdigit(c)</td>
<td>True if c is a digit</td>
</tr>
<tr>
<td>islower(c)</td>
<td>True is c is a lowercase letter</td>
</tr>
<tr>
<td>isupper(c)</td>
<td>True if c is a uppercase letter</td>
</tr>
</tbody>
</table>

You can refer to this webpage for more details about `<cctype>`
http://www.cplusplus.com/reference/cctype/?kw=cctype
• **Question:** Will this program compile? If so, what’s the output?

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

int main () {
    string c = 'hi5';
    cout << isalnum(c[2]) << endl;
}
```

Output:
**Question:** Will this program compile? If so, what’s the output?

```cpp
#include <iostream>
#include <string>
#include <cctype>

using namespace std;

int main () {
    string c = "hi5";
    cout << isupper(c[2]) << endl;
}
```

**Output:**

0

isupper(c) returns non-zero value if c is an uppercase alphabetic letter. Zero otherwise.
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.

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

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

```cpp
#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)

```cpp
#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

```cpp
#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:

At line 9:

```cpp
cin               inputInt
3' '2' '\n'
```

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

```
cin               inputString
\n'
```

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.
Stream

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

int main () {
    string inputString;
    int inputInt;
    cout << "Enter a number: ";
    cin >> inputInt;
    cout << "Input was: " << inputInt << endl;
    cin.ignore(10000, 'n');
    cout << "Enter a string: ";
    getline(cin, inputString);
    cout << "Input was: " << inputString << endl;
}
```

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`.

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

Input:
32
world

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

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

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

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

```java
if (conditional1){
    // Execute if conditional1 is evaluated as true
    statement1;
    statement2;
    ...
} else if (conditional2){
    // Execute if conditional1 is evaluated as false AND
    // conditional2 is evaluated as true
    ...
} else {
    // Execute if conditional1 is evaluated as false
    statementn;
}
```

```java
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.

<table>
<thead>
<tr>
<th>symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“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?

<table>
<thead>
<tr>
<th>Boolean statement</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \geq y )</td>
<td>False</td>
</tr>
<tr>
<td>( x = y )</td>
<td>False</td>
</tr>
<tr>
<td>( x - y &gt; 10 )</td>
<td>False</td>
</tr>
<tr>
<td>( x &gt; y ) &amp;&amp; ( y &lt; z )</td>
<td>False</td>
</tr>
<tr>
<td>( (x \neq y)</td>
<td></td>
</tr>
<tr>
<td>( x \neq y)</td>
<td></td>
</tr>
<tr>
<td>( x \neq y)</td>
<td></td>
</tr>
</tbody>
</table>
If-else statements

• **Question:** Will this program compile? If so, what’s the output?

```cpp
#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.
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 or enum).
- It is something similar to concatenating if-else statements, but limited to constant expressions.

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

**Switch example**

```cpp
switch (x) {
    case 1: cout << "x is 1";
        break;
    case 2:
        cout << "x is 2";
        break;
    default:
        cout << "value of x unknown";
}
```

**If-else equivalent**

```cpp
if (x == 1) {
    cout << "x is 1";
} else if (x == 2) {
    cout << "x is 2";
} else {
    cout << "value of x unknown"
}
```
Switch statements

• **Question:** Will this program compile? If so, what’s the output?

```cpp
#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";
    }
}
```

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x is 1</td>
</tr>
<tr>
<td>2</td>
<td>x is 2 or 3</td>
</tr>
<tr>
<td>3</td>
<td>x is 2 or 3</td>
</tr>
<tr>
<td>4</td>
<td>x is not 1, 2 nor 3</td>
</tr>
</tbody>
</table>
Switch vs. if-else

• https://www.quora.com/What-is-the-difference-between-switch-and-if-else-statements-Which-is-better-in-different-cases-Why
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.

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

Don’t forget the ‘;’ here.
Loops

• **Question:** Will this program compile? If so, what’s the output?
Loops

• **Question:** Will this program compile? If so, what’s the output?

```cpp
#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?

```cpp
#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.

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

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

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

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

```cpp
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:

```
N
N
NN    N
N
N
N    NN
N
N
```

Hints:

\[
\begin{align*}
\text{for } i = 0 \text{ to } 4 & : \text{print 'N'} \\
\text{for } i = 5 & : \text{do nothing} \\
\text{for } i = 6 \text{ to } 9 & : \text{print 'N'}
\end{align*}
\]
Programming challenge

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

int main () {
    // Assume SIZE must be odd
    const int SIZE = 5;
    // For loop that goes through
    // SIZE rows...
    for (int i = 0; i < SIZE; i++) {
        // And loop that goes through
        // SIZE columns
        for (int j = 0; j < SIZE; j++) {
            // [!] Condition for printing 'N'
            // HINT: 3 cases for printing one!
            if (j == 0 || j == 4 || j == i) {
                cout << "N";
                // Otherwise, print a space
            } else {
                cout << " ";
            }
        }
        cout << endl;
    }
}
```
Advice

• When encountering new problems in C++,
  • Google
  • www.cplusplus.com
  • www.stackoverflow.com
Credit to 2 previous CS31 TAs

• This slide is finished with reference from:
  • Andrew Forney
  • http://web.cs.ucla.edu/~forns/
  • Brian Choi
  • http://netlab.cs.ucla.edu/~schoi/cs31/
Thanks!