CS 162
Intro to Computer Science II

Lecture 1

Review C++ Basics

1/10/24

Oregon State University

Odds and Ends

- Assignment 1 posted
- Check discord -> notes & resources
  - VS code setup
  - Bypass DUO
  - C/C++ for python programmer
  - C++ tutorial
  - Map a network drive
  - Linux commands

Lecture Topics:

- Review C++ Basics:
  - Data type and Variables
  - Program Input/Output
  - Control Structure
    - If/else
    - Loops
  - Functions
    - Pass by value vs. Pass by reference
  - 1D static array

C++ Program Structure

- `main()` function: -- entry point into the program
- Include statements at the top of the file
  - i.e., #include <iostream>

```
#include <iostream> //standard I/O, writing to / reading from the console/file


int main() {
        return 0;
}
```

1 Byte = 8 bit 1 or 0 for a bit

C++ Primitive Types

- **Data type:** define how the computer's memory is allocated and how data is stored and manipulated
- **Primitive data types in C++: char, double, float, int, long, short, bool**
- Fundamental
  - **short/int/long**: whole numbers, e.g. 45, -89, 0
  - **float/double**: real numbers, e.g. 2.612, -30.5, 2.3e5
  - **char**: a single character, e.g. 'A', '&', 'x', '\'
  - bool: boolean, e.g. true, false (all lower case)
  - string: e.g. "A", "hello world"
- **Signed** by default – can represent both positive and negative values
- **Unsigned** – represent non-negative values only (i.e., double the range of positive values)
  - **unsigned int, unsigned short, unsigned long**

Variables

- Variable – Memory location with name and type to store value
- Variable declaration – Statement requesting variable w/ name and type
  - Examples:

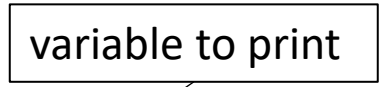
    double height;

    int age;
- Declare and initialize a variable in the same statement:
  - int value = 5;
- Variable/identifier name: Start with letter(upper-case, lower-case, underscore (_)), followed by sequence of letters and digits
  - Good examples: hiThere, two_plus_two, _hello
  - Bad examples: 5dogs, hi-there, hello there
  - Can't use keywords

Program Input/Output

- C++: **cout**
  - Example:

```
std::cout << "The integer value is: " << value;
```



  - What about the newline?
    - `'\n'` or `std::endl`

- C++: **cin**
  - Example:

```
std::cin >> value;
```

C++ If/else and switch statements

```
if (a == 0) {
  /* Do something. */
}
else if (b != 0) {
  /* Do something different. */
}
else {
  /* Do a third thing altogether.
*/
}
```

```
switch(grade) {
        case 'A' :
                cout << "Excellent!" << endl;
                break;
        case 'B' :
        case 'C' :
                 cout << "Well done!" << endl;
                break;
        case 'D' :
                 cout << "You passed!" << endl;
                break;
        case 'F' :
                 cout << "Try again!" << endl;
                break;
        default :
                 cout << "Invalid grade!" << endl;
}
```

C++ Loops

- C++:
  - for, while, do-while

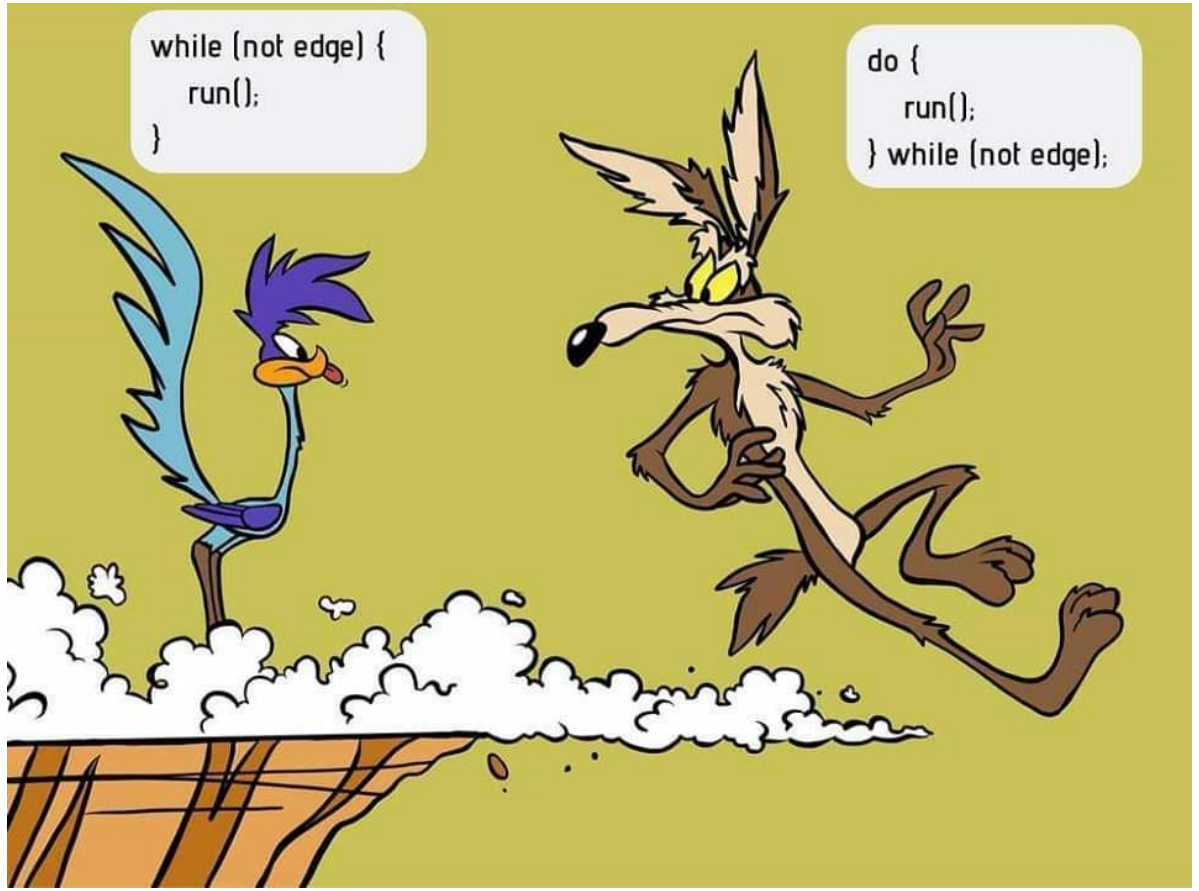
```
int i;
for (i = 0; i < 32; i++) {
  /* Do something 32 times. */
}
```

```
while (i != 16) {
  /* Do something repeatedly until i is 16. */
}
```

```
do{
  /* Do something repeatedly until i is 16. */
}while (i != 16);
```

Recap: loops


A WIFE CALLS HER PROGRAMMER HUSBAND AND TELLS HIM "WHILE YOU'RE OUT, BUY SOME MILK."
HE NEVER RETURNS HOME.
imgflip.com


while (not edge) {
run();
}
do {
run();
} while (not edge);

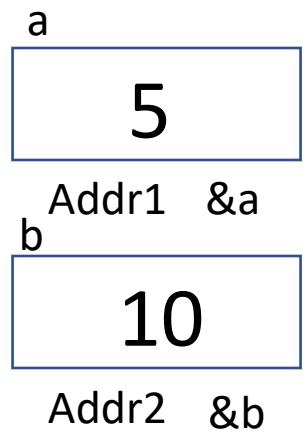
C++ Functions

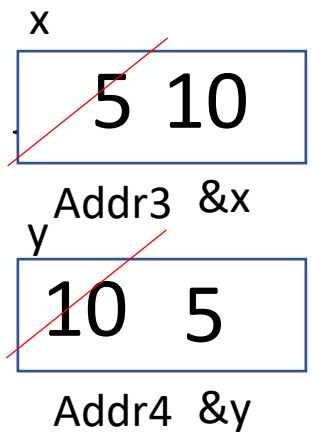
```cpp
#include <iostream>

using namespace std;

float cal_avg(float num1, float num2);

int main()
{

    float total = 0;
    float count = 0;

    cout << "Enter total: ";
    cin >> total;
    cout << "Enter count: ";
    cin >> count;

    float average = cal_avg(total, count);
    cout << "Avg.: " << average << endl;

    return 0;
}

float cal_avg(float num1, float num2){
    return num1/num2;
}
```

- Label:
  - Function declaration/prototype
  - Function call
  - Function definition
  - Function name
  - Parameter(s)
  - Argument(s)

C++ Pass by Value

```
void swap(int, int);
int main() {
    int a=5, b=10;
    swap(a, b);
    cout << "a: " << a << "b: " << b;
}
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
```





C++ References

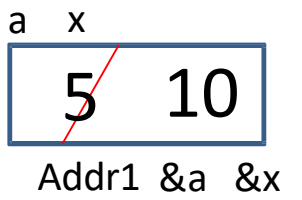
- Reference == a variable that refers to a particular memory address
- Reference declaration: `int i = 4; int &i_ref = i;`
  - A reference MUST be initialized
  - Once initialized, the memory address referred to by a reference variable can't change
    - i.e. `i_ref` above must always refer to the address of `i`.
    - Quick check: what will the following code print?

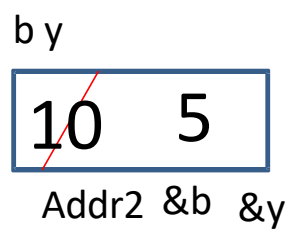
```
int a = 7, b = 2, &ref = a;
cout << a << " " << ref << endl; // prints
ref = b;
cout << a << " " << ref << endl; // prints
```

-> Trying to make a new assignment to a reference changes its value

C++ Pass by Reference

```
void swap(int &, int &);
int main() {
   int a=5, b=10;
   swap(a, b);
   cout << "a: " << a << "b: " << b;
}
void swap(int &x, int &y) {
   int temp = x;
   x = y;
   y = temp;
}
```



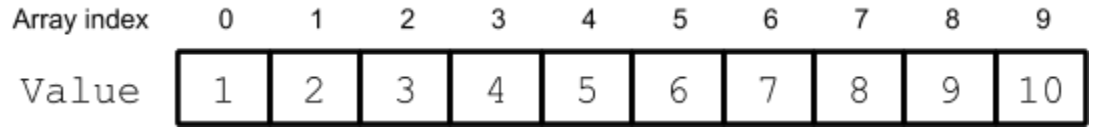


1D static Arrays

- An array is a contiguous block of memory holding values of the same data type
- Static Arrays: created on the stack and are of a fixed size, during compiling time
  - 1-dimensional static array: `int stack_array[10];`
    - You can initialize an array at the same time as you declare it:

      `int array[] = {1,2,3,4,5,6,7,8,9,10};`

      Note: you can omit the size if you initialize the array when you declare it
    - Array name: stores the starting address of the array
    - i.e., array == &array == &array[0]
    - Conceptually, the array above looks like this:



C/C++ Pointers

- Pointers == variables that hold memory addresses
- Variable declaration: `int a = 5;`
  - Creates a variable on the stack of size int with the value 5
- Pointer declaration: `int *b = &a;`
  - Creates a pointer variable on the stack which can hold an address of an int and sets the value of the pointer (the address the pointer points to) to the address of `a`
- Dereferencing Pointer: `cout << *b << endl;`
  - Dereference: access the value stored in the memory address held by a pointer
  - Will print the value stored at the address which `b` points to
- Every pointer points data of a specific data type

C++ Pointers

```
void swap(int *, int *);
int main() {
    int a = 5, b = 10;
    swap(&a, &b);
    cout << "a: " << a << "b: " << b;
}
void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}
```