CS 161
Intro to CS I

1-D Arrays, Strings, and Command Line Arguments
Chap. 5
C++ Pass by Value

```cpp
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;
}
• What if we didn’t have temp?
```

---

OSU Oregon State University
C++ Pass by Reference

```cpp
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++ 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;
}
Variables vs. References

• Value Semantics
  – Values stored directly
  – Copy of value is passed
  int i, j=2;
i=j;

• Reference Semantics
  – References to the value are stored
  – Copy of address is passed
  int *i, j=2;
i=&j;
What is an Array?

- **Array (ar·ray) n.** An ordered arrangement of related items.
  - Example: Array of colors in a rainbow.
    - Related items?
    - Ordered arrangement?
  - Class examples?
  - Computer Science
    - Same data type/data structure
    - Contiguous memory locations
Create 1-D Array

```c
int student_grades[5];
```

- How do you access each item?
- What does the array name represent?
- Why is the array name the address of 1\textsuperscript{st} element?
- What are the initial values?
Initialize/Assign Values

• **Declaration**
  
  ```c
  int student_grades[5] = {0, 0, 0, 0, 0};
  ```

• **Individual Elements**
  
  ```c
  student_grades[0]=0;
  ...
  student_grades[4]=0;
  ```

• **Why is this incorrect?**
  
  ```c
  student_grades={0, 0, 0, 0, 0};
  ```
Initialize/Assign Values...

• Using a Loop
  While Loop Example:
  i=0;
  while (i<5) {
    student_grades[i]=0;
    i++;
  }

  For Loop Example:
  for(i=0; i<5; i++)
    student_grades[i]=0;

• Which is better to use with arrays and why?
Read/Print 1-D Array Values

• Read Values From User

```cpp
for(i=0; i<5; i++) {
    cout << "Enter final grade for student: ";
    cin >> student_grades[i];
}
```

• Print Values

```cpp
for (i=0; i<5; i++) {
    cout << "Student\'s final grade is " << student_grades[i] << endl;
}
```
```cpp
#include <iostream>
#include <cstring>

using std::cout;
using std::cin;
using std::endl;

void print_intro(char name[]) {
    cout << "Hello, " << name << endl;
}

int main() {
    char name[20];

    cout << "Enter your name: ";
    cin.getline(name, 20);

    cout << "Address of name & 1st element: " << (void *) &name[0] << endl;
    cout << "Address of 2nd and 3rd elements: " << (void *) &name[1] << endl;
    cout << "First letter in name and addr: " << name[0] << endl;
    cout << "Length: " << strlen(name) << endl;

    // How do we print the last letter?
    cout << "Last letter in name and addr: " << name[strlen(name)-1] << endl;

    // What happens when we step off the array bounds?
    // cout << "What if we go a little off: " << name[21] << endl;
    // cout << "What if we go way too far: " << name[100000] << endl;

    print_intro(name);
}
```
```
#include <iostream>
#include <cstring>

using std::cout;
using std::cin;
using std::endl;

void print_intro(char name[]){
    cout << "Hello, " << name << endl;
}

// Notice, we have argc and argv as parameters in main
int main(int argc, char *argv[]){
    char name[20];

    cout << "The number of arguments is 1 or greater: " << argc << endl;
    cout << "The first argument is the program name: " << argv[0] << endl;
    cout << "The second argument is one that you supply: " << argv[1] << endl;

    /* cout << "Enter your name: ";
    cin.getline(name, 20);
    cout << "Address of name & 1st element: " << (void *) name 
```