Introduction to C++

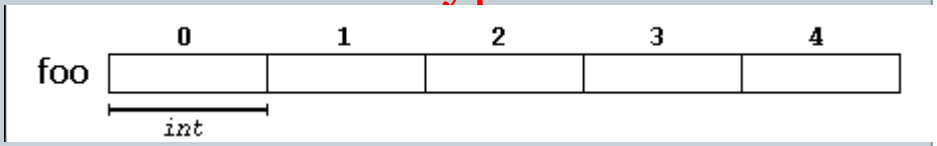
Outline

- Arrays
- Character Sequences
- Pointers

Arrays

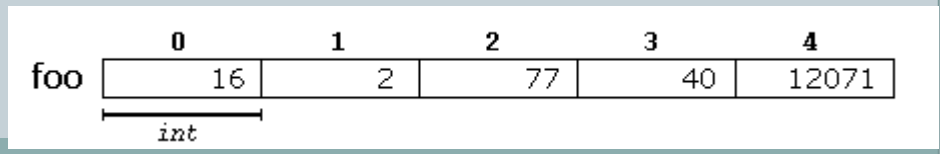
- Arrays in C++ are similar to Python lists in some ways, but there are significant differences
- To declare an array
  - You need to specify the data type AND the size
  - All items in the array must be the SAME data type

```
int foo [5];
```



- To declare and initialize:

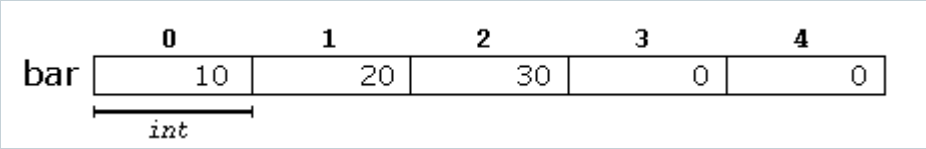
```
int foo [5] = { 16, 2, 77, 40, 12071};
//OR
int foo [] = { 16, 2, 77, 40, 12071};
```



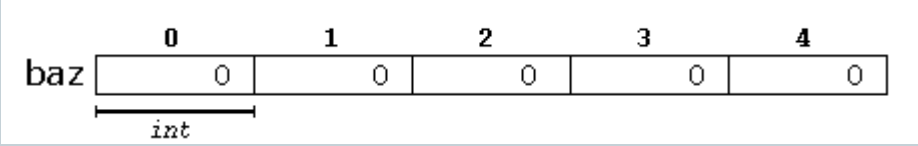
Arrays

- If you declare and array and initialize it with fewer values than specified, the remaining values will be set to the default for that data type

```
int bar [5] = { 10, 20, 30 };
```

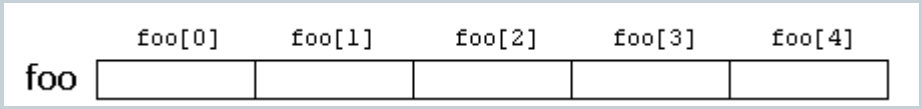



```
int baz [5] = { };
```



Accessing Array Values

- Just like in Python, provide the name[index]:




```
foo[2] = 75;
x = foo[2];
```

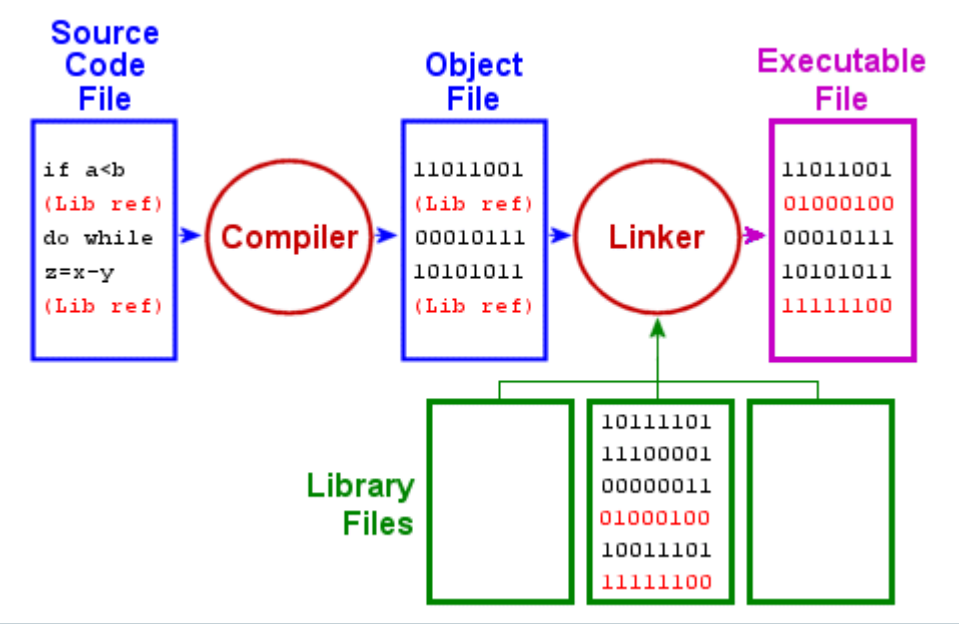
- What will happen in the following code?

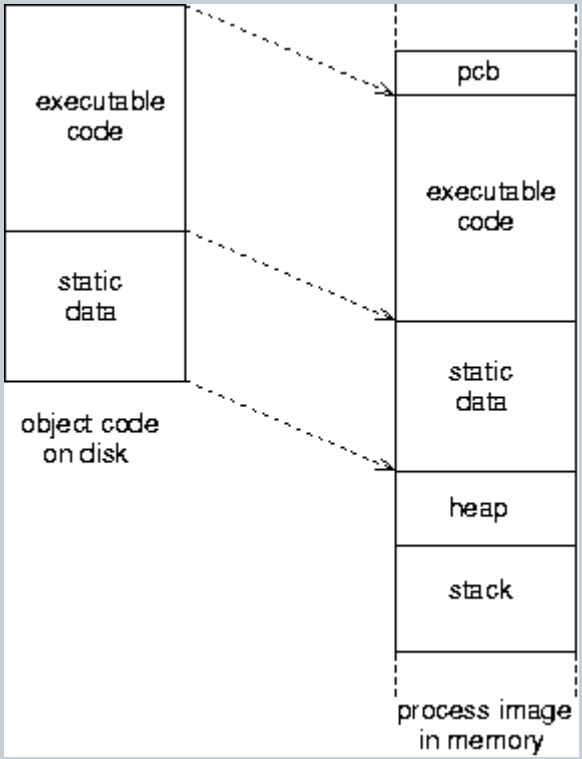
```
int foo [5] = {16, 2, 77, 40, 12071};
foo[6] = 10;
cout << foo[6] << endl;

10
16 2 77 40 12071 32765 10
```

Accessing Array Values

- What just happened?!?

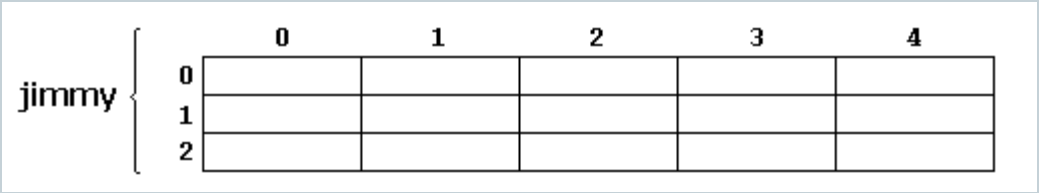




Multidimensional Arrays

- Arrays of arrays

```
int jimmy [3][5];
```



- Just a programming convenience – data is still stored contiguously in memory – the following are stored the same way:

```
int jimmy [3][5];
int jimmy [15];
```

Passing Arrays as Parameters

- You can pass an array as a parameter to a function
  - Array is not copied – only a pointer to the array is passed
  - Pass by reference

```
void someFunction(int arr[])
    {
        …
    }

int myArray [40];

someFunction(myArray);
```

Passing Arrays as Parameters

- You can also pass multidimensional arrays as a parameter
  - First dimension is left empty

```
void someFunction(int arr[][3][4])
    {
        ….
    }

int myArray [40][3][4];

someFunction(myArray);
```

Character Sequences

- A string is really just a sequence (array) of characters

```
char foo [20];
```

- You can use this to assign values:

```
char myWord[] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

- Or, C++ allows you to assign a string directly during initialization:

```
char myWord[] = "Hello";
```

  - C++ will put the null character in the array automatically

- Note: this won't work in subsequent code – you'll need to assign values individually

Character Sequences

- Strings and character arrays can be used interchangeably with cin and cout
- But – arrays have a fixed size while strings have no defined size

```
// strings and NTCS:
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  char question1[] = "What is your name? ";
  string question2 = "Where do you live? ";
  char answer1 [80];
  string answer2;
  cout << question1;
  cin >> answer1;
  cout << question2;
  cin >> answer2;
  cout << "Hello, " << answer1;
  cout << " from " << answer2 << "!\n";
  return 0;
}
```

```
What is your name? Homer
Where do you live? Greece
Hello, Homer from Greece!
```

Character Sequences

- You can convert between the two:

```
char myntcs[] = "some text";
string mystring = myntcs;  // convert c-string to string
cout << mystring;          // printed as a library string
cout << mystring.c_str();  // printed as a c-string
```

Summary

- Arrays
- Character Sequences
- Pointers