Pointers and Dynamic Memory

The pointer concept

- A pointer variable holds the address of a data value
  - it does **NOT** hold actual data
  - declared in C++ by placing a * in front of a variable
  - default value is null (does not “point” to anything)

- address of regular variables can be found using &
  - not to be confused with pass by reference!!

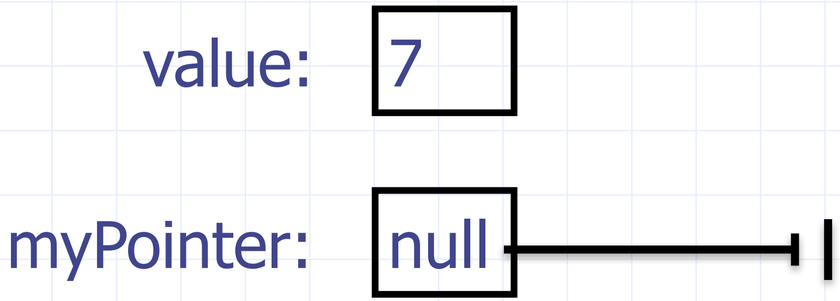
- to “dereference” a pointer variable, prefix it by a *

Simple Pointer Example

```
int value=7;
int *myPointer; // pointer declaration
myPointer = &value;
cout << value << endl; // 7
cout << *myPointer << endl; // 7
*myPointer = 34;
cout << *myPointer << endl; // 34
cout << value << endl; // 34 (!!!!)
```

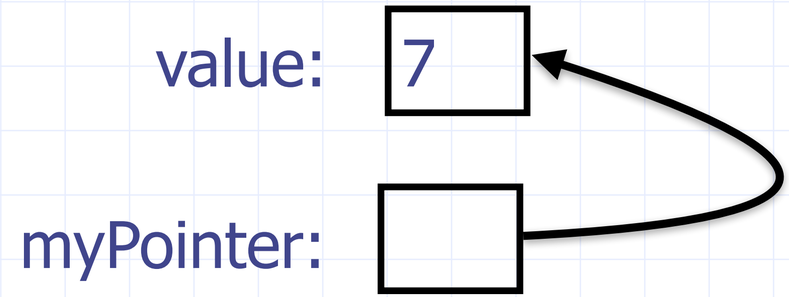
Simple Pointer Example ...

```
int value=7;
int *myPointer; // pointer declaration
myPointer = &value;
cout << value << endl; // 7
cout << *myPointer << endl; // 7
*myPointer = 34;
cout << *myPointer << endl; // 34
cout << value << endl; // 34 (!!!!)
```



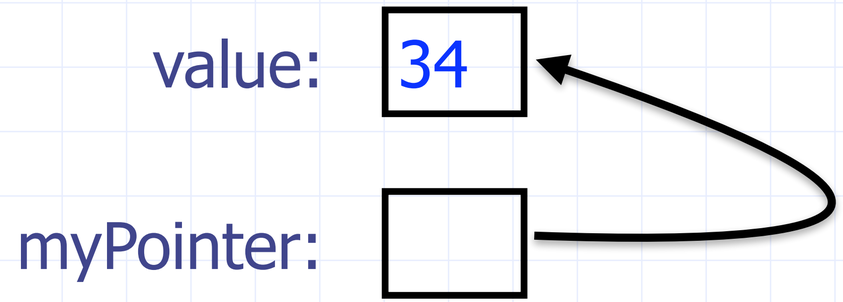
Simple Pointer Example ...

```
int value=7;
int *myPointer; // pointer declaration
myPointer = &value;
cout << value << endl; // 7
cout << *myPointer << endl; // 7
*myPointer = 34;
cout << *myPointer << endl; // 34
cout << value << endl; // 34 (!!!!)
```



Simple Pointer Example ...

```
int value=7;
int *myPointer; // pointer declaration
myPointer = &value;
cout << value << endl; // 7
cout << *myPointer << endl; // 7
*myPointer = 34;
cout << *myPointer << endl; // 34
cout << value << endl; // 34 (!!!!)
```



Dynamic Allocation

- You can use the **new** keyword in C++
  - just as in Java, allocates new memory and …
  - … returns a reference to such
  - unlike Java, you can allocate space for intrinsic types
    - int, char, double, etc …
- exs.

```
int *iPtr = new int;
double *dPtr = new double;
char *cPtr = new char; // special meaning, more later
Complex *cpxPtr = new Complex(2, 6);
```

Classes and Pointers

- Suppose you had just coded:

```
Complex *cpxPtr = new Complex(2, 6);
```

- To call the getReal() method from the Complex class

```
double realPart = (*cpxPtr).getReal(); // () needed!
```

- Syntax is considered ugly … so ugly, there is an alternative:

```
double realPart = cpxPtr->getReal();
```

  - Note the arrow (->) … used with an (object) pointer variable!

Classes and Pointers and **this**

- The keyword **this** in a class is a pointer to the current object

- The += operator, which should really return an appropriate object
  - Complex a, b(2,3), c(4,5);
  - should be able to do : a = b += c; // a= (b+=c);

```
const Complex&
Complex::operator+=(const Complex & other)
{
   _real += other._real;  //could do   this->_real = …
   _imaginary += other._imaginary;
   return *this; // “this” is not a local variable!!!!
}
```

Arrays and Pointers

- A pointer can "point to" an array:

```
int myArr[100]; // 100 element int array
int *aPtr = myArr;
```

- Can now do the following:

```
for (int index; index<100; index++)
   cout << aPtr[index] << endl;
```

- Can also allocate arrays dynamically:

```
aPtr = new int[75]; //aPtr now a new 75 element array
```

  - Note that this does not impact `myArr` !!!

Pointer Arithmetic

- Consider the following code:

```
int myArr[100];
int *aPtr = myArr;
// … code to put values in array is omitted here
for (int i=0; i<100; i++)
    cout << *(aPtr+i) << ' ';
cout << endl;
```

- Or even …

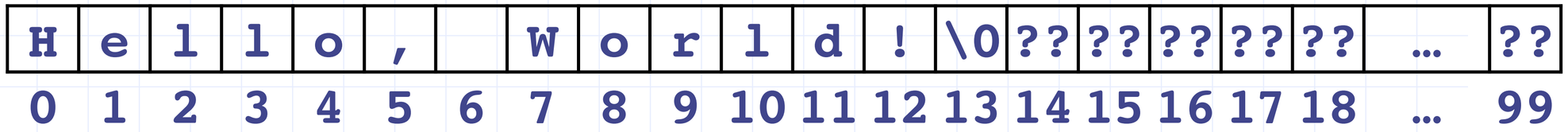
```
for (int i=0; i<100; i++)
   *aPtr++ = 100-i; // stores {100, 99, …, 1} in array!
```

Character Pointers and C-Strings

- A pointer to char can point to a character array

```
char *chPtr = new char[100];
```

- A character array can be be thought of as a C-String:
  - comes from C programming language, had no “string” type
  - all characters from beginning of array are in the string until …
    - … null (‘\0’, equivalent to integer 0) is encountered



C/C++ main() Method

- One of the valid headers for the main method is:

```
int main(int argc, char* argv[])
```

- The second argument (argv) is:
  - an array of …
  - char* (i.e. c-strings)
  - so, the second argument is an array of strings!
    - sound like Java to you?
- The first argument (argc) is the size of the array
  - i.e. number of strings.

- each string is a command line argument (program name is first)