

# For Friday

- Read Savitch chapter 10 carefully
- C++ Practice Program 3
  - Assignment is described on Blackboard.
  - Submit through Blackboard **from the Suns**.
  - Start Firefox on the Suns by typing `firefox&` at any command prompt.

# Program Quality

# String I/O

- Printing strings is very straightforward
- Reading them can be more of an issue
- `>>` will read one “word”
- To read a whole line, use `getline`
  - Member function for C-strings
  - Function that accepts a stream and a string parameter for string class strings

# Character I/O

- get
- put
- putback
- peek
- ignore

# Using Characters

- There are a number of built-in functions for manipulating or answering questions about characters.

# string class

- Very similar to Java's in many ways
- Benefits from C++ operator overloading (=, ==, and []) all work as expected)
- Use `c_str` to convert to C-string

# Operator Overloading

- Really just a function.
- Often may be either member function or not.
- Name of the function is operator followed by the symbol.
- Parameters are generally const reference.

# Commonly Overloaded

- ==
- <<
- >>
- Others depend on the kind of thing

# When to Overload

# Friend Functions

- What?
- Why?
- How?

# Pointers

- Value is a memory address
- Similar to references
- Can manipulate the value of the address directly
- Must explicitly dereference the pointer to access the thing being referred to (or pointed at)

# Declaring Pointers

```
int *p, *q;
```

```
char *str;
```

```
Student *stuPtr;
```

# Notes on Pointers

- In a multi-variable declaration, the \* is required for each individual variable
- The \* is not part of the name of the variable; it is part of the type
- Each pointer has an associated type, called the target type
- The target type does not affect the actual value of the variable, but it does affect C++'s manipulation of the variable

# Values of Pointers

- Pointers are not automatically initialized to anything
- You must place an address in them
- Size of pointers varies based on the capabilities of the machine
- Often pointers are the same size as longs

# The Address Operator

- Allows us to determine the address of an item in memory.

# The Address Operator

```
int v, *ptr;
```

```
double x, *y;
```

```
ptr = &v;
```

```
y = &x;
```

# Dereferencing Pointers

- Use the asterisk before the pointer variable

```
v = *ptr;
```

# Comments

- Do not dereference a pointer unless the address points to an appropriate value
- NULL is usually used as the value to indicate a pointer that is pointing to nothing
- NULL is actually the 0 value pointer

# Practice with Pointers

- Write a declaration-initialization to establish a pointer, `charPtr`, to a location that stores a character and place the letter 'B' in that location. Declare any other variable necessary.

# Arrays and Pointers

- What **is** an array name?
- If we declare

```
int    numArray[10];
```

what is the value of numArray?

# Arrays and Pointers

- The name of an array is a constant pointer
- That is, `numArray` is the address of `numArray[0]`

# Using Pointers As Arrays

```
int scoreArray[10];  
int *scorePtr = scoreArray;  
*scorePtr = 10;  
scoreArray[1] = 12;  
scorePtr[2] = 11;
```

# Addresses of Array Elements

- Address of array elements are computed by adding the address of the first element to the product of the size of the elements and the element index
- Thus in an integer array `ar` starting at address `100`, the address of `ar[2]` would be `100 + 2 * sizeof(int)` or `108` if we assume 4 byte integers

# You Try It

- We have the following declarations:  
`char carr[10];`  
`long larr[20];`
- Assume chars are 1 byte and longs are 4 bytes.
- The address of carr is 240 and the address of larr is 320
- What is the address of carr[6]?
- What is the address of larr[4]?

# Pointer Arithmetic

- We can actually do arithmetic (addition and subtraction) with pointers
- This works very much like the computation to find an element in an array
- You take the current value of the pointer and add the number being added times the size of the target data type

# Pointer Arithmetic Practice

- Given:

```
int *ptr1;  
char *ptr2;  
float *ptr3;
```

- What is:

```
ptr1 + 5  
ptr2 + 11  
ptr3 + 3
```

# Final Note

- For any array **a**:

**a[i] == \*(a+i)**