Announcements

• Make sure each of you join the Google Group created for CMPSC 24 (both Lecture Sections):
  – http://groups.google.com/group/cs-24-spring-2010

Lecture outline

• Discussion of pointers

• Recursive Data Structures:
  – Linked Lists
Definitions

• **An address** is a location in memory:
  – All data (variables) in a program have addresses
• A **pointer** is a variable that stores “address” of other data/variables.

(pointer variable)

Definitions

• **Dynamic Data** is memory that is allocated within your program while the program is executing, i.e., at run-time:
  – Does not have name
  – Run-time system dynamically assigns an address based on the allocation
  – Access needs to managed via pointer variable that stores the address
  – The pointer has a name but the memory location it points to does not have a name
  – Program must de-allocate the data when no longer needed

(Pointer Types)

• **Pointer variable**
  • A variable whose value is the address of a location in memory

(pointer variable)

Pointer Types

```c
int alpha;
int* intPointer;
intPointer = &alpha;
```

If alpha is at address 33, memory looks like this:

```c
  1  2  3  4  5  6  7  8
```

Because ptr holds the address of x, we say that ptr "points to" x.

```
  x
```

Pointer Types

```c
int x;
x = 12;
int* ptr;
ptr = &x;
```

Because ptr holds the address of x, we say that ptr "points to" x.

```
  x
```

Pointer Types

- **Dereference operator (*)**
  - An operator that, when applied to a pointer variable, denotes the variable to which the pointer points.

- **Dynamic allocation (new operator)**
  - Allocation of memory space for a variable at run time (as opposed to static allocation at compile time).
```
Pointer Types

```int x;
x = 12;
``` int* ptr;
ptr = &x;
```std::cout << *ptr;
``` Because ptr holds the address of x,
we say that ptr “points to” x.
*ptr is the value in the place to which ptr points
```
```}

```
Pointer Types

```int x;
x = 12;
``` int* ptr;
ptr = &x;
```*ptr = 5;
``` // changes the value of x to 5
```

```
Pointer Types

```char ch;
ch = 'A';
``` char* q;
q = &ch;
```*q = 'Z';
``` char* p;
p = q;
``` // now p and q both point to ch
```
```
```
Dynamic allocation

```plaintext
int* intPointer = new int;
```

Pointer Types

- **NULL Pointer**
  - A pointer that points to nothing
- **Memory Leak**
  - The loss of available memory space that occurs when memory is allocated dynamically but never deallocated
- **Garbage**
  - Memory locations that can no longer be accessed

Why Pointers?

- Used in manipulating dynamic data structures:
  - Data-structures that grow and shrink over time
  - E.g., Arrays and Strings using pointers
  - Linked structures
Why Pointers?

• Used to share data:
  – Two different data objects need to access the same piece of data

STUDENT: Jane Doe

Why Pointers?

• Optional Data:

Computer A

Printer

Computer B

Printer

Computer A

Computer B
Linked List Using Pointers

struct NodeType {
    int info;
    NodeType* next;
};

NodeType* list;
NodeType* locaLon;

locaLon -> info is the same as (*locaLon).info

Linked List

Be sure you understand the differences among
location, *location, and location->info

location -> info is the same as (*location).info
Linked List Operations

- Set list to empty
- Add a new node to list
- Count the number of nodes in list
- Remove the last node from list