CS 161
Intro to CS I

Pointers
Introduction

• Definition
  • **Pointer** is the address of a variable in memory

• You may remember them from call-by-reference parameters
Pointer Variables

• Pointer is a type
  • Pointer variable holds a pointer, i.e. an address
  • NOT int or double or other numeric data type!

• Pointer variable declaration
  • double *ptr;
  • Pointer to double
  • Holds pointer to only type double
  • Why?
Pointer “Jargon”

• Pointers hold and address. But...
  • We don’t mention the address

• A pointer ‘points’ to a variable of the correct data type

• More on these shortly

• Dereferencing a pointer is getting the value stored in the location

• Address of operator allows us to manipulate the address
Pointing

int *ptr1, *ptr2, var1, var2;

ptr1 = &var1;
   & is the address of operator
   Statement sets ptr1 to point to var1

Translation
   ptr1 points to var1 also means
   ptr1 equals address of var1 or
   ptr1 holds the address of var1
Pointing

Consider-

```c
int *ptr1, *ptr2, var1, var2;
ptr1 = & var1;
```

Two ways to get the value of var1
```c
var2 = var1;
var2 = *ptr1;
```

* is the **dereferencing operator**
Retrieves the value ptr points to (i.e. that storage location)
Example

Consider-
  
```cpp
var1 = 0;
ptr1 = &var1;
*ptr1 = 42;
cout << var1 << endl;
cout << *ptr1 << endl;
```

Produces output:
  
```
42
42
```

ptr1 and var1 refer to same variable
Assigning Pointers

Pointer values can be assigned to other pointers

```c
int *p1, *p2;
p1 = p2;
Assigns pointer 2 to pointer 1
P1 now points to where p2 points to
```

Consider this-  *p1 = *p2;
What’s different?

Value in p2 is placed in p1
The pointers (i.e. the addresses) do not change
Static

Memory allocated at compile time

Variable = Value Semantics
Variable name associated to memory location
Value stored
Copy of value is used

```
int i, j = 2;
i = j;
```
Dynamic

Memory allocated at runtime
Memory comes from the heap

Pointer = Reference Semantics
Allocate memory
Address stored in pointer variable

```c
int *i = NULL;
i = new int;
*i = 2;
```
Memory Leak

...  
int main () {
    int *i=NULL; //created in main function
    while(1) {
        i = new int;
    }
}

Memory allocated in the heap and not released
Plugging the Leak

...  
int main () {
    int *i=NULL; //created in main function
    while(1) {
        i = new int;
        delete i; frees or releases the memory
    }
}

Memory allocated in the heap is now available to use elsewhere in your program
Arrays

• Pointer is the address of the first array element

Array1[2] = 3AE17 + 2* 20_{16} = 3AE57 (address of element 3)

• You may also remember them from call-by-reference parameters: int foo(int &x);
Suggestion

Is this confusing? YES

It will never make sense unless you write programs and use pointers!

It will never make sense unless you write programs and use pointers!