Week 3 Lecture 3

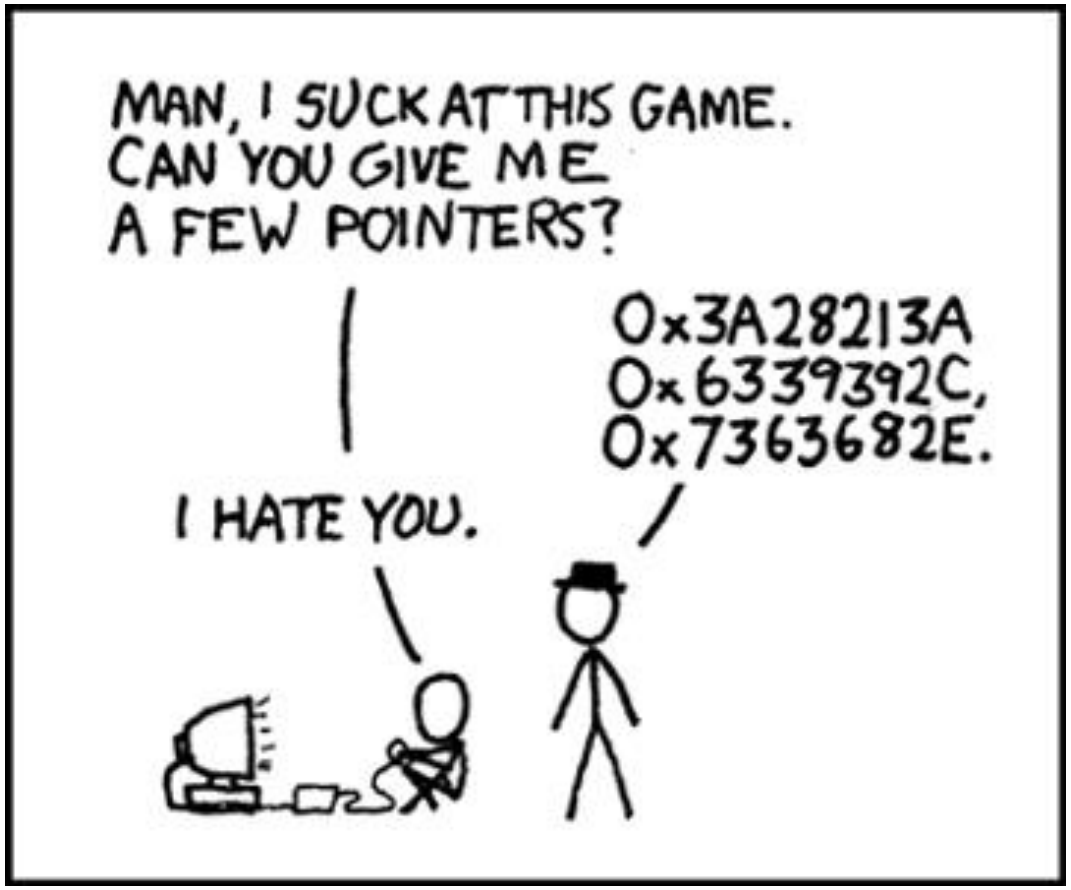
NWEN 241
Systems Programming

Jyoti Sahni

Jyoti.sahni@ecs.vuw.ac.nz

Content

- Pointers



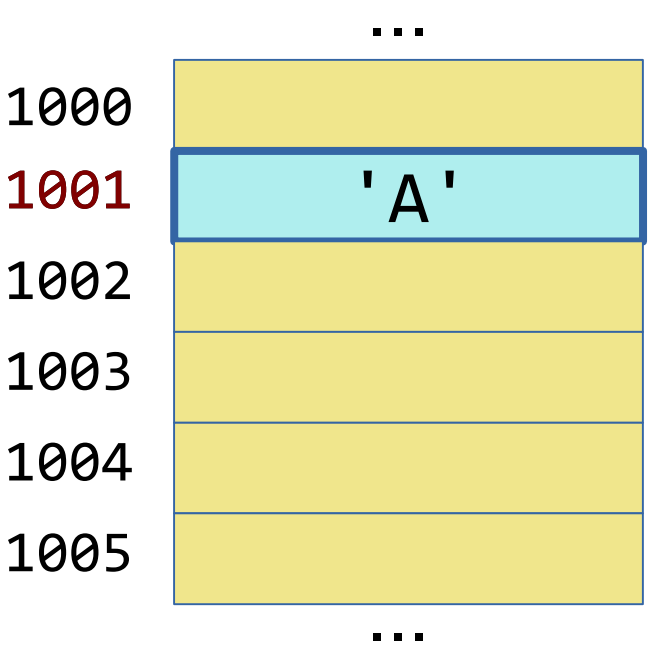
Recap: Memory Location and Variables

- A variable declaration allocates memory to store the value of the variable

```
char c = 'A';
```

Memory location 1001 contains value of variable c

A variable directly references a value



Recap: Memory Location and C

- C provides the ability to access specific memory locations, using **pointers**

Pointers are variables that contain memory addresses as their values

Variable vs Pointer

A variable directly references a value

A pointer indirectly references a value

Declaring a Pointer

- Pointers are typed based on the type of entity that they point to
  - To declare a pointer, use * preceding the variable name as in:

```
data_type *name;
```

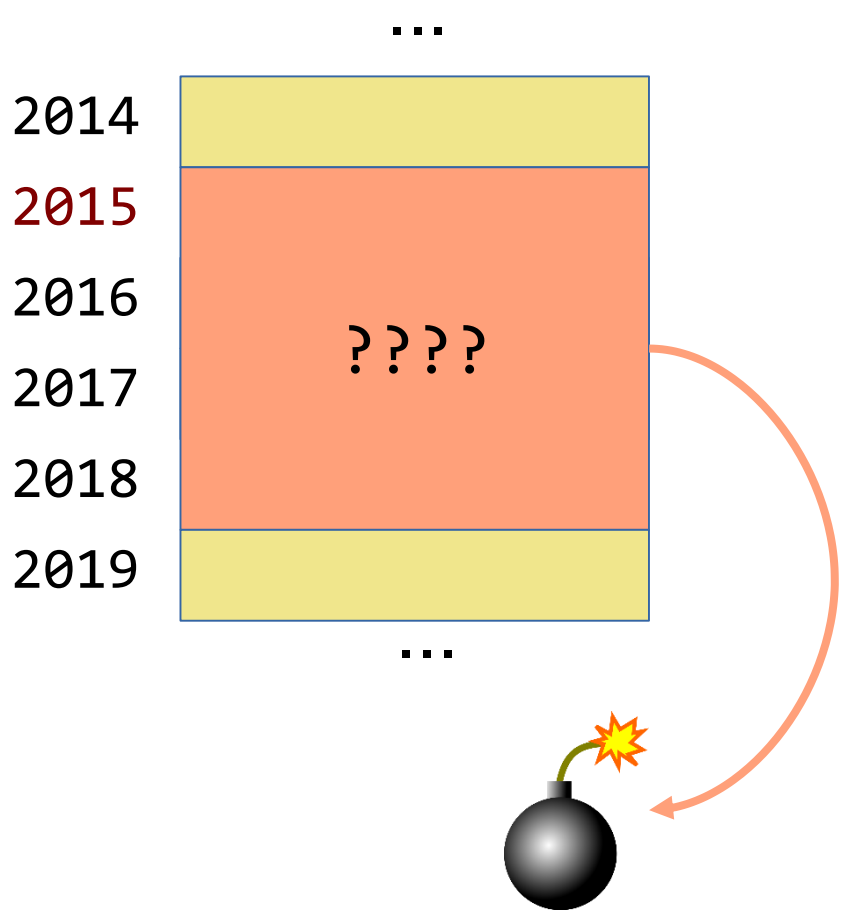
  - Examples:

```
int *p;    // p is a pointer to an int
float *q;  // q is a float pointer
char *r;   // r is a char pointer
int *s[5]; // s is an array of 5 int pointers
```

What Happens in a Pointer Declaration?

```
int *p;
```

- Memory is allocated that can store an address
- The size of this space depends on the number of bits used for addressing
- The initial contents may be some ‘rubbish’ number
  - This means the pointer may point to arbitrary memory locations

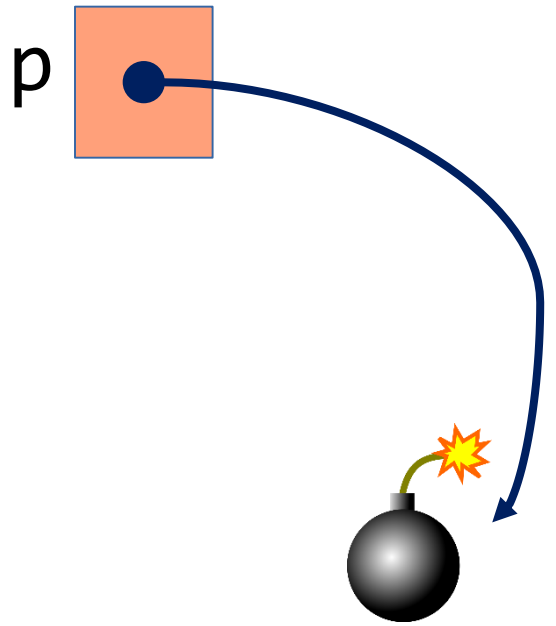


What Happens in a Pointer Declaration?

```
int *p;
```

- Memory is allocated that can store an address
- The size of this space depends on the number of bits used for addressing
- The initial contents may be some ‘rubbish’ number
  - This means the pointer may point to arbitrary memory locations

Simplified graphic to illustrate pointers:



Address Operator (&)

- The address (&) operator can be used in front of any variable
  - The operation will return the memory location of the variable

```
&name
```

name can be any ordinary variable or even a pointer variable

- Example:

```
int a, *x;
x = &a;
/* x variable contains address of a, i.e.,
x points to variable a */
```

```
int a, *x;
x = &a;
/* x variable contains address of a, i.e.,
x points to variable a */
```

Indirection Operator (*)

- A pointer variable contains a memory address
- To refer to the *contents* of the variable that the pointer points to, we use indirection operator

```
*name
```

name is a pointer variable

- Example:

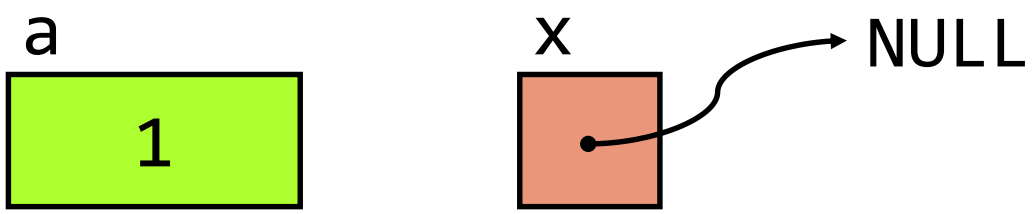
```
int a = 100, b, *x;
x = &a;
b = *x;
/* b will be assigned the content pointed
to by x, which is 100 */
```

```
int a = 100, b, *x;
x = &a;
b = *x;
/* b will be assigned the content pointed
to by x, which is 100 */
```

Graphical Illustration

Declaration:

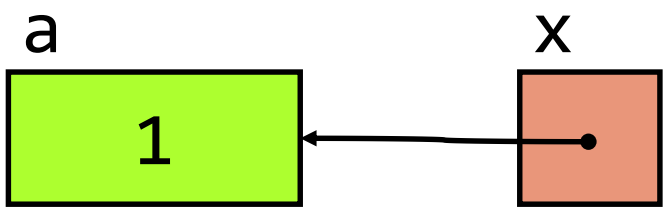
```
int a = 1; int *x = NULL;
```



NULL – pointer literal/constant to non-existent address

Assignment:

```
x = &a;
```



Pointer Basics (1)

- Given:

```
int i = 10, *p;
p = &i;
```

- Suppose `i` is at address 100 and `p` is at address 200.
- What is `i`?
- What is `&i`?
- What is `*i`?
- What is `p`?
- What is `&p`?
- What is `*p`?

Pointer Basics (2)

- Given:

```
int a = 1, b = 5; int *x;
x = &a;     // What is the value of x ?
*x = *x + 1;      // a = __ ;  b = __ ;
b = *x;
```

- What is the value of b ?

Usage of Pointers

1) Provide an alternative means of accessing information stored in arrays

2) Provide an alternative (and more efficient) means of passing parameters to functions

3) Enable dynamic data structures, that are built up from blocks of memory allocated from the heap at run time

Next Lecture

- More Pointers
- Storage Classes