CSE 30: Computer Organization and Systems Programming

Lecture 6: C Data types: Pointers

Diba Mirza
University of California, San Diego
## C Integer Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>4 bytes</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
</tr>
<tr>
<td>long</td>
<td>4 bytes</td>
</tr>
<tr>
<td>long long</td>
<td>8 bytes</td>
</tr>
<tr>
<td>unsigned</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>
Other datatypes

```c
char c='a';  /* 1 byte */
```

- By default, may be signed or unsigned
- Always enforce using the appropriate keyword

```c
float f;    /* 4 bytes */
double d;   /* 8 bytes */
```
int x, y; /* What is the value of x?*/
y = 10;
x = 20;
y = x > y ? x : y;
Pointers

• **Pointer:** A variable that contains the *address* of a variable
• **Declaration:**  *type*  * *pointer_name*;

```c
int  *x;
```

How do we initialize a pointer?
Accessing location

To access the location/address, use the address operator ‘\&’

```c
int *x;
int y = 20;
```

Pointers

• How to make a pointer point to something.

```c
int *x;
int y;
y = 3;
x = &y;
```
Pointers

• How to make a pointer point to something?

```c
int *x, y;
y = 3;
x = &y;
sizeof(x) =
```

![Diagram showing the process of making a pointer point to something.](attachment:diagram.png)
• Short hand diagram for the following scenario

\[
\begin{array}{c}
\text{x} & 120 \\
\text{y} & 3 \\
\end{array}
\]

102
To change the value of a variable using pointers: use dereference \( * \) operator to left of pointer name.

```c
int y=3, *x;

x = &y;

*x = 5;
```

Two ways of changing the value of any variable
Why this is useful will be clear when we discuss functions and pointers

Change the value of y directly:

Change the value of y indirectly (via pointer x):
Pointers and Pointees

int *p1, *p2, x;
p1 = &x;
p2 = p1;

Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.  

B.  

C. Neither, the code is incorrect
Q: This code gives a warning at compile time. Why?

```c
char *p;
int y;
p = &y;
```

A. The pointer ‘p’ is made to point to a variable of incompatible type

B. *p does not contain a valid value because y was not initialized
Q: What happens when we run the following code?

```c
int *p;
*p = 5;
```

A. `p = 5`
B. `*p = 5`

C. Compile time error
D. Runtime error
Segmentation faults (aka segfault)

• Indicates that your program has crashed!
• What caused the crash?
  – Segfaults occur if your program is trying to read or write an illegal memory location.
Q: What is the output of this code?

```c
int *p, x = 5;
p = &x;
printf("%d",(*p)++);
```

A. The value pointed to by p, which is 5
B. The value pointed to by p plus one, which is 6
C. Undefined
D. Compiler error
E. Segmentation fault
Two important facts about Pointers

1) A pointer can only point to one type – (basic or derived) such as `int`, `char`, `struct`, another pointer, etc.

2) After declaring a pointer: `int *ptr;`
   `ptr` doesn’t actually point to anything yet. We can either:
   - make it point to something that already exists, or
   - allocate room in memory for something new that it will point to
   - Null check before dereferencing