Pointers!

A pointer is a variable that contains the address of a variable.

1. Pointers and Addresses

   \[ p = \& c; \]
   \[ \ast p = \ast (\& c) = c \]

   `show live code!`

   \& - address of operator. \( p = \& c; \)

   \[ \Rightarrow \text{applies only to objects in memory.} \]
   \[ \Rightarrow \text{eg. variables and array elements.} \]
   \[ \Rightarrow \text{cannot be applied to.} \]
   \[ \text{- expressions} \]
   \[ \text{- constants} \]
   \[ \text{- register variables.} \]

   \* - indirection or dereferencing operator.
Pointers and Function Arguments

* C passes arguments to functions by value.
* **WRONG** swap method!
* **CORRECT** swap function. - using pointers!

```c
void swap(int *px, int *py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
```

Diagram:
- In caller:
  - `a`:
  - `b`:
- In swap:
  - `px`:
  - `py`:
Pointers and Arrays

```c
int a[10];
```

```
int *pa;
pa = &a[0];
```

```
x = *pa;
```

Diagram illustrating the memory layout of `a` and how `x` is accessed using the pointer `pa`.
\( \ast (a+i)^4 \equiv a[i] \)

Applying \& operator on both sides.

\& (\ast(a+i)) \equiv \&(a[i])

\[ \Rightarrow a+i \equiv a[i] \]

if \( pa \) is a pointer,

then \( pa[i] \equiv \ast(pa+i) \)

\( \equiv \) an array-and-index expression

\( \equiv \) a pointer and offset

\( \ast \) Diff bet array name and pointer names.

\( pa = a; \) \( \checkmark \)

\( pa++; \) \( \checkmark \)

\( a = pa; \) \( \times \)

\( a++; \) \( \times \)
Passing array names to functions.

```c
len = mystrlen(a);

int strlen(char *s)
{
    int n;
    for (n = 0; *s != '\0'; s++)
        n++;
    return n;
}
```

pass a part of an array.
NULL pointer

int *p = NULL;

#define NULL 0

Pointer Arithmetic / Operations

0. Assignment of pointers of same type: \( q. p = q \)
1. Add/Subtract a pointer and an integer
   \( p = p + 1; \quad p = p - 1; \)
2. Subtract or compare 2 pointers (same array)

\[
\begin{align*}
0x100 & \quad 101 & \quad 102 & \quad 103 & \quad 104 & \quad 105 \\
\end{align*}
\]

\[
n = q - p + 1 = 104 - 101 + 1
\]

\( p < q \) \checkmark

3. Assigning or comparing to 0
   \( p = 0; \checkmark \quad p = \text{NULL}; \) (preferred) \checkmark
Illegal pointer operations (p, q - pointers)

1. p+q  X
2. p*q  X
3. p/q  X
4. p + 7.5  X
5. int xpn = &n;
   char int xpc = &c;
   pn = pc;  X
   pc = (char *) pn;  ✓

↓ casting an int* to a char *
character pointers

char amessage[] = "Let it be";
char *pmassage = "Hey Jude";

amessage: \textit{Let it be\textbackslash 0}

pmassage:
\texttt{Hey Jude\textbackslash 0}

\textit{can change \texttt{pmassage} to point to a different string.}

\underline{String copy}

\underline{Array version

\texttt{void strcpy(char *s, char *t)}

\{'
\ \ \ \ \ \ int i;
\ \ \ \ \ \ i = 0;
\ \ \ \ \ \ while ((s[i] = t[i]) != \textbackslash 0)
\ \ \ \ \ \ \ \ \ \ \ \ i++;
\\texttt{\'}}
Pointer version

```c
void stripy (char *s, char *t)
{
    while ((*s != *t) || (*s == '0')) {
        s++;
        t++;
    }
}
```

Advanced:
```c
void stripy (char *s, char *t)
{
    while ((s++ != t++) || (*s == '0')) ;
}
```

More advanced:
```
while (*s++ = *t++)
;```
To Read in K&R C:

chap 5.1 to 5.5 !

Lecture 0 - CS APP: 1

1 - K&R chap 2 (except bitwise operators).

2 - K&R chap 7 (7.1, 7.2, 7.4, 7.5, 7.7).

Project 0 - due Friday.

scanf

BIOS