

ICS-104

Pointers

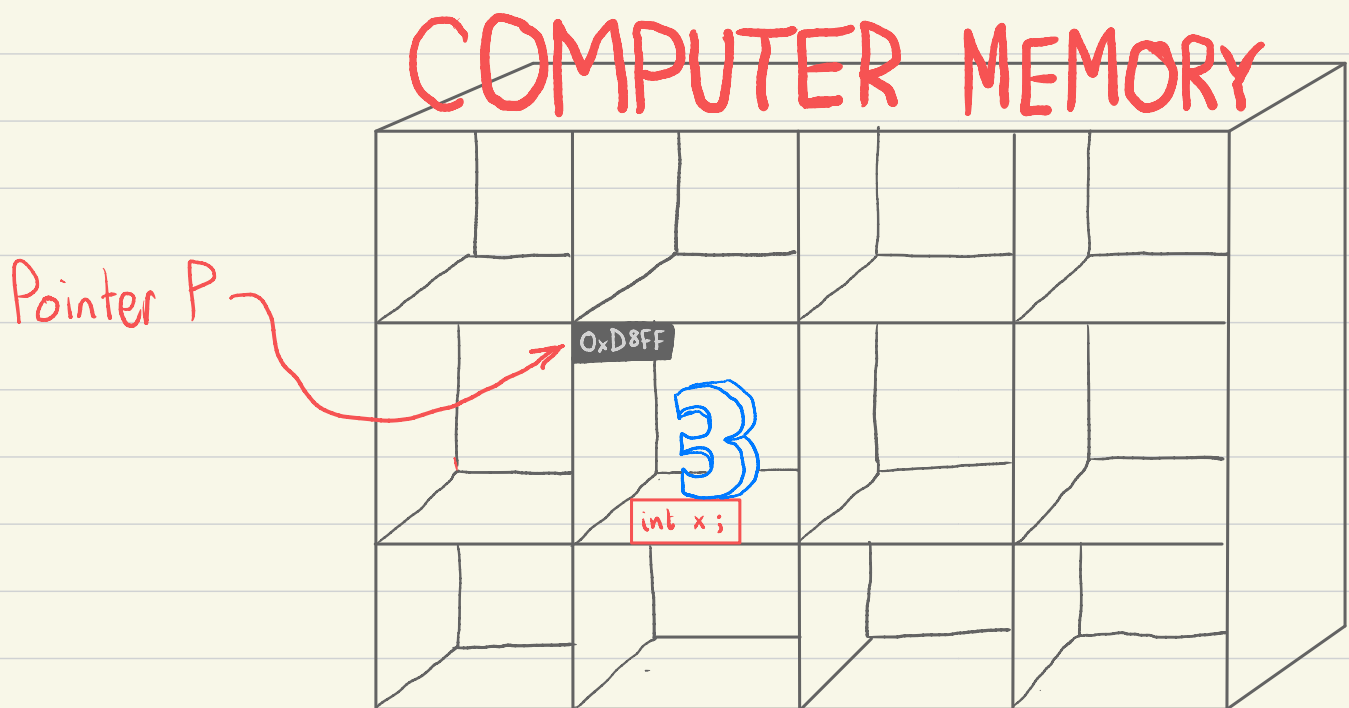
By: Naif Alqahtani

 Twitter: @NaifAlqahtani

 Youtube.com/MWNLLT

• What are pointers?

- When we call a variable, we usually want to use the value of that variable. However, sometimes, we want to know the LOCATION of that variable in memory and NOT its value
- Pointers are like arrows that point to the location of the variable in memory.
- Imagine a storage room where there are shelves
 - The shelves represent the memory.
 - Each cabinet represent a memory location
 - Each variable is stored inside a cabinet



Here we have a variable 3 called x stored in memory location 0xD8FF and a pointer P that stores the address of variable x.

• Pointers in C

- In the previous example, we created a variable `x` and set it to 3.
- Suppose we want to create another variable that access that same value exactly and not a copy of it.
- we first create a pointer like so:

```
int *p;
```

- Here the star means give me an arrow to a location of type int.
- Then we code:

```
p = &x;
```

 - Here we say, set the pointer to point (not to `x`) but to the ADDRESS of `x` by using `&` which means give me the location and not the value (3)
- This means the value of `p` will be `0xD8ff` and not 3.
- Therefore, we establish that `*` means pointer and that `&` means address of.

• Also, we now know how to declare a pointer.

```
double *pointer;
```

Diagram illustrating the declaration of a pointer:

- `double` is labeled as "type of pointer".
- `*` is labeled as "name of pointer".
- `pointer` is labeled as "pointer".

• File Pointers in C

- To open use a file in C, we need:
 - A Pointer
 - The **FILE** prefix.

• Example:

FILE ^{pointer} ***input**;
 ↓ ↓
 type of pointer pointer name

- Then we can use **fopen()** to open a file.
- Notice the **f**
- The Function is similar to python.

input = **fopen**("input.txt", "r");
 ↓ ↓ ↓
 pointer reference name of file mode

• To read the file, we use **fscanf()**.

- Requires 3 arguments: File pointer, Format, variables
- Returns the number of successful values read, or -1 if end of file (EOF) is reached

fscanf(**input**, "%d", &**x**);
 ↓ ↓ ↓
 opens this file Reads as int Stores the value in x

• To write to the file, we use **fprintf()**.

- Requires 3 arguments: File pointer, Format, variables
- Returns the number of successful values written, or -1 if it fails.

fprintf(**output**, "%d", 2020);
 ↓ ↓ ↓
 File to write to in "w" mode type of character to write place holder variable

• To close a file we use **fclose()**

- **fclose(input);**

• Reading and writing from a file— Example:

input file: indata.txt

```
344 55 6.3556 9.4
43.123 47.596
```

```
#include <stdio.h>

int
main(void)
{
    FILE *inp;           /* pointer to input file */
    FILE *outp;          /* pointer to output file */
    double item;         /* variable buffer */
    int input_status;    /* status value returned by fscanf */

    /* Prepare files for input or output */
    inp = fopen("indata.txt", "r");
    outp = fopen("outdata.txt", "w");

    /* Input each item, format it, and write it */
    input_status = fscanf(inp, "%lf", &item);
    while (input_status == 1) {
        fprintf(outp, "%.2f\n", item);
        input_status = fscanf(inp, "%lf", &item);
    }

    /* Close the files */
    fclose(inp);
    fclose(outp);

    return (0);
}
```

file pointer

variable buffer

status variable

open files with correct modes

read

write

repeat until EOF is reached

print value to file

read next value

store current value in variable item

read first char to set status value

close files

return exit code

TABLE 6.4 Different Kinds of Function Subprograms

Purpose	Function Type	Parameters	To Return Result
To compute or obtain as input a single numeric or character value.	Same as type of value to be computed or obtained.	Input parameters hold copies of data provided by calling function.	Function code includes a return statement with an expression whose value is the result.
To produce printed output containing values of numeric or character arguments.	void	Input parameters hold copies of data provided by calling function.	No result is returned.
To compute multiple numeric or character results.	void	Input parameters hold copies of data provided by calling function. Output parameters are pointers to actual arguments.	Results are stored in the calling function's data area by indirect assignment through output parameters. No return statement is required.
To modify argument values.	void	Input/output parameters are pointers to actual arguments. Input data is accessed by indirect reference through parameters.	Results are stored in the calling function's data area by indirect assignment through output parameters. No return statement is required.

• Scope of names:

- The **scope of a name** refers to the region of a program where a particular meaning of a name is visible or can be referenced.
- The scope of:
 - constant macros begins at their definition and continues to the end of the source file.
 - a function subprogram begins with its prototype and continues to the end of the source file.
 - formal parameters and local variables extends from their declaration to the closing brace of the function in which they were declared.
- A local variable or a formal parameter may block the definition of a function (e.g.) if it has the same name.

تمنياتى لكم بالتوفيق والنجاح
واختبار ميسر بإذن الله
لاتنسوني من صالح الدعوات

 Twitter : @NaifAlqahtani

 Youtube.com/MWNLLT