Pointers in C

By: Yehia M. Abu Eita

Outlines

- Introduction
- Pointer declaration
- Pointer arithmetic
- Arrays Vs. pointers

Introduction

- Pointer is a **non-primitive** data type that **stores the address** of another already existing variable.
- Pointer **size depends** on the platform, depending on **address bus size**.
- In embedded systems you **can not access memory mapped registers without pointers**.
- Without pointers there is **no way** to make **call-back** functions in embedded systems.

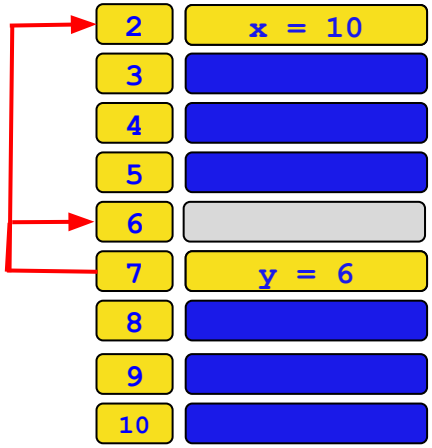
Pointer declaration

- **Declaration example:**
  - `int *ptr; // ptr is a pointer to integer`
- **Definition example:**
  - `int *ptr = &x; // prt now points to x`
  - **If address of x is 2, then ptr will store 2**
- **Accessing variables through pointers:**
  - You can **change** variable's value **indirectly** using the **' * ', dereferencing operator**.
  - **Example:** `*ptr = 10; // will change the value of x to 10.`

Pointer declaration

- **Declaration examples:**
  - `int *x; // x is a pointer to integer`
  - `int **x; // x is a pointer to pointer to integer.`
  - `int *x[10]; // x is an array of 10 pointers to integer.`
  - `int (*x)[10]; // x is a pointer to an array of 10 integers.`
  - `int *(*x)[10]; // x is a pointer to an array of 10 pointers to integer.`
  - `int (*x)(int, int); // x is a pointer to a function that takes two integers and returns one integer.`
  - `int *(*x)(int, int); // x is a pointer to a function that takes two integers and returns a pointer to an integer.`

Pointer arithmetic



- Some arithmetic operations can be done on pointers:
  - **Addition (+):** operates on **all types of pointers except void pointers.**
  - **Subtraction (-):** operates on **all types of pointers except void pointers.**
  - **Increment (++):** operates on **all types of pointers except constant and void pointers.**
  - **Decrement (--):** operates on **all types of pointers except constant and void pointers.**
- Pointer arithmetic operation result **depends on the size** of the data the **pointer points to**.

```
int x = 5, *y;
y = &x;
*y = 10;
y++; // y = y + 1;
```

Arrays Vs. pointers

- **Similarities:**
  - **Arrays** can use pointer's expressions, ***(x+i)** and **x+i**.
  - **Pointers** can use array's expressions, **ptr[i]** and **&ptr[i]**.
  - **Some** of **arithmetic operations** have the same results in both pointers and arrays, **+ and -**.
- **Differences:**
  - **Array name is not a pointer**, array name **is the address of the first element in the array.**
  - **The arithmetic operators ++, and -- cannot be used with arrays.**
  - Pointers are **variables** that **store addresses** of **other** variables.
  - **Array size** is the **number of elements** reserved for the array.
  - **Pointer size** depends on the **address bus**.

Summary

- Now you are more familiar with pointers.
- You can easily declare, define, reading pointer declarations using the SOAC technique.
- Remember, pointer arithmetic depends on data you point to.
- Take care of the differences between arrays and pointers.