Pointers

In ‘C’

The Concept

- Variable's value can be addressed by their name, or by their address (their locations)
- In 'C', this is done by using pointers
- A Pointer is Address of a Variable and not it's value.
- The concept is well understood at assembly language level.

Pointers Cont.....

- In low level assembly language special registers are used as pointers.
- Variables are addressed (pointed to) by these special registers called ‘Index Registers’ or ‘ Address Registers’.
- In Intel (8085, 8086) registers ‘H’ & ‘L’ and registers ‘D’ & ‘E’ are paired to form a 16 bit pointers.

Pointers Cont……

- In Motorola Processors namely 68000 registers A0 to A7 are or can be used as address registers in an index mode.
- E.g.

```
move.b  (A0), D0 ;
```

This means move the data pointed by register A0 to a Data register D0

The address mode for this operation is known as ‘Indexed’

Pointers Representation in ‘C’

- If ‘x’ is a variable and ‘px’ is a pointer
- Using an ‘&’ Operator gives address of ‘x’ to ‘px’
- i.e px=&x meaning ‘px’ is an address and a ‘pointer to’ the value of variable ‘x’
- The operator ‘&’ can only be applied to a variable or array elements
- & (x+1) and &5 are illegal syntax

Pointers Representation in ‘C’

- The un array operator is **‘*’**
- **‘*’** Treats the operand as the address of the ultimate target.
- E.g. **z= *px**

means whatever **px** is pointing to is assigned to **‘z’**

px=&x ; **z=*px;** both are the same as : **z=x;**

Declaration of pointers in ‘C’

Int x,z;

Int *px;

***px** is intended as mnemonics

Which means **‘px’** is an integer, if it appears in the context of ***px**.

Syntax of the declaration for a variable mimics the syntax of expressions in which the variable might appear

Examples

double sort(), *ep

Meaning **sort() , *ep** have type double.

Pointers can occur in expressions

e.g. If ***ep** is a pointer for **x** then ***ep** can occur in any context that **x** can.

Y=*ep+1 is equivalent to **y=x+1 or**

println ("%d\n",*px)

The expression prints the current value of **x**

d=sqrt((double)*px)

Produces in **'d'** the square root of **x** corrected into **double** before it is passed to **sqrt**

• **'%'** in **'C'** means **modulus** operator
e.g. **X % Y** Produces the remainder when **x** is divided by **y.**

'%' can not be applied to **float** or **double**

• Pointer reference can also occur on the left side of the assignment.
e.g. If px points to x
Then ***px=0** sets x to zero
***px+=1** sets x to one
(*px)++ increments x
Brackets are needed to increment **x**

Otherwise ***px** is incremented

Since pointers are **variables** then they can be manipulated as other variables.

If py is another pointer then **py=px** copies the contents of **px** into **py**

i.e. **py** points to whatever **px** was pointing to.