C PROGRAMING

by Dr.Akshita Chanchlani @ Sunbeam Infotech

Pointer

Pointer - Introduction

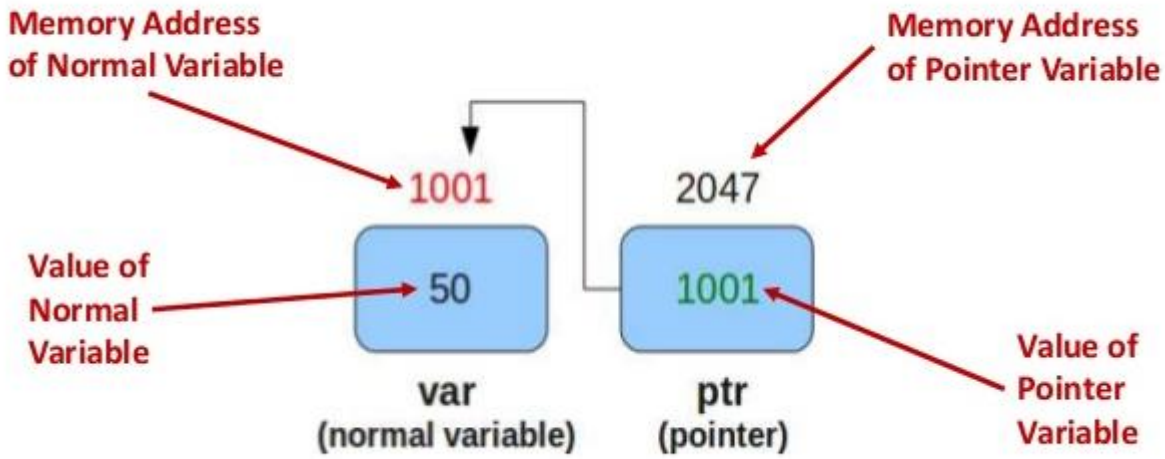
- Pointer is a variable that stores address of some memory location.
- Internally it is unsigned integer (it is memory address).
- Using pointer concept we can create a variable to store address.
- In C, pointer is a special data type.
- It is not compatible with unsigned int.
- Pointer is derived data type (based on primitive data type).
  - To store address of int, we have int pointer.
  - To store address of char, we have char pointer, …
- Size of pointer variable is always same, irrespective of its data type (as it stores only the address).

Pointer - Syntax

- Pointer syntax:
  - Declaration:
    - int *p;
  - Initialization:
    - p = &i;
  - Dereferencing:
    - printf("%lf\n", *p);
- Reference operator - &
  - Also called as direction operator / address of
  - Read as “address of”.
  - used to get address of variable
- Dereference operator - *
  - Also called as indirection operator or deference or value at operator
  - Read as “value at”
  - used to get value at the address stored in a pointer

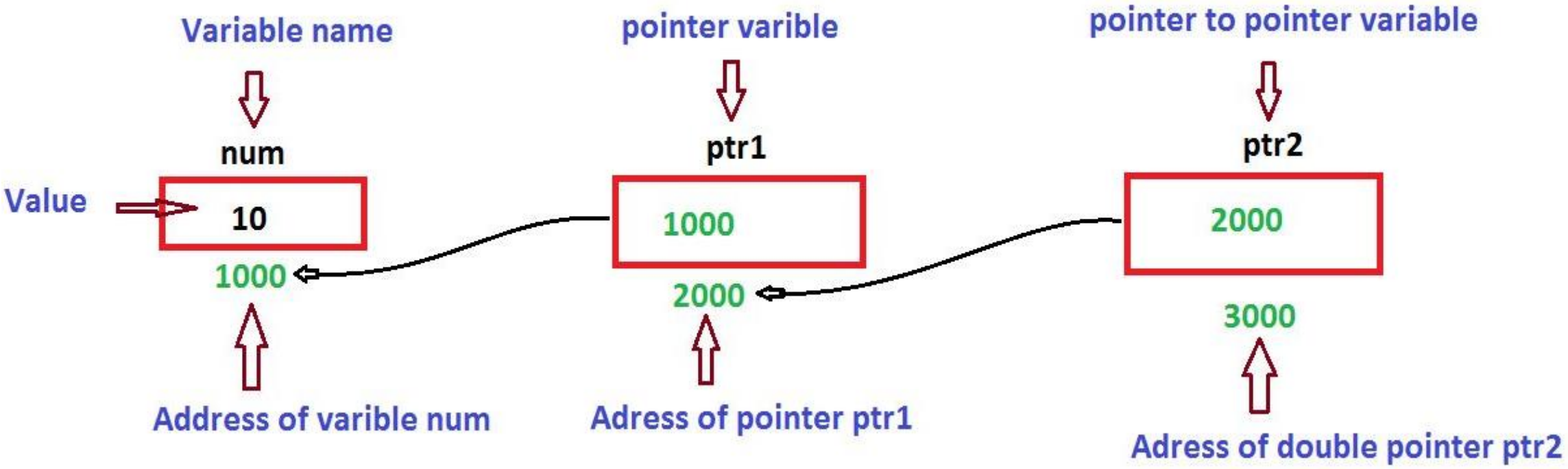
Example of Pointer

```
void main() {
    int var = 50;
    int *ptr;           /*Declaration*/
    ptr = &var;         /*Initialization*/
    printf("%d",*ptr);/*(50) accessing value*/
    *ptr = 20;                  /*setting value*/
    printf("%d",var);/*(20)*/
}
```



Level of Indirection

- We can declare n indirection level of pointers as shown in diagram.



Void pointer and null pointer

- void * is a pointer which is designed to hold address of any location. It is a generic pointer.
- Example :
  - char *cptr=100 //assumes address is of character whose scale factor is 1byte
  - *cptr derefers 1 byte from given base address 100

- Examle:
  - int *ptr=100; //assumes address is of integer whose scale factor is 4 bytes
  - *ptr derefers 4 bytes from given base address 100

- void pointer is not aware of scale factor i.e. how many bytes to be dereferred from given base address. Hence if we want to receive value at void pointer it is necessary always typecast prior its to use.

void pointer

- Void pointer is generic pointer it can hold address of any data type (without casting).
- Scale factor of void* is not defined, so cannot perform pointer arithmetic.
- To retrieve value of the variable need type-casting.
- void* is used to implement generic algorithms.

NULL pointer

- If pointer is uninitialized, it will hold garbage address (local pointer variables).
- Accessing such pointer may produce unexpected results. Such pointers are sometimes referred as wild pointers.
- C defined a symbolic const NULL, that expands to (void*)0.
- It is good practice to keep well known address in pointer (instead of garbage).
- NULL is typically used to initialize pointer and/or assign once pointer is no more in use.
- Many C functions return NULL to represent failure.
  - strchr(), strstr(), malloc(), fopen(), etc.

Pointer and ++

- A pointer in c is an address, which is a numeric value.
- Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value. There are four arithmetic operators that can be used on pointers: ++, --, +, and -.
- Icrementing a pointer Variable Depends Upon data type of the Pointer variable.
- new value = current address + i * size_of(data type)

Three Rules should be used to increment/decrement pointer :

1. Address+1=Address

2. Address++ = Address

3. ++Address = Address

Pointer - Scale factor

- Size of data type of pointer is known as Scale factor.
- Scale factor defines number of bytes to be read/written while dereferencing the pointer.
- Scale factor of different pointers
  - Pointer to primitive types: char*, short*, int*, long*, float*, double*
  - Pointer to pointer: char**, short**, int**, long**, float**, double**, void**

Pointer arithmetic

- Scale factor plays significant role in pointer arithmetic.
- n locations ahead from current location
  - ptr + n = ptr + n * scale factor of ptr
- n locations behind from current location
  - ptr - n = ptr - n * scale factor of ptr
- number of locations in between
  - ptr1 – ptr2 = (ptr1 – ptr2) / scale factor of ptr1

Pointer to function

- Pointer as a function parameter is used to hold addresses of arguments passed during function call.
- This is also known as **call by reference**. When a function is called by reference any change made to the reference variable will effect the original variable.

Functions returning Pointer variables

A function can also return a pointer to the calling function.
In this case you must be careful, because local variables of function doesn't live outside the function. They have scope only inside the function. Hence if you return a pointer connected to a local variable, that pointer will be pointing to nothing when the function end

Thank you!

Dr.Akshita Chanchlani <akshita.chanchlani@sunbeaminfo.com>