Pointers

Dr. B.S.R.V. Prasad
Department of Mathematics
School of Advanced Sciences
Vellore Institute of Technology
Vellore - 632014, TN, India

What are Pointers?

- A pointer is a derived data type in c.
- Pointers contains memory addresses as their values.
- A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location.
- Like any variable or constant, you must declare a pointer before using it to store any variable address.
- Pointers can be used to access and manipulate data stored in the memory.

Advantages

- Pointers make the programs simple and reduce their length.
- Pointers are helpful in allocation and de- allocation of memory during the execution of the program.
  Thus, pointers are the instruments dynamic memory management.
- Pointers enhance the execution speed of a program.
- Pointers are helpful in traversing through arrays and character strings. The strings are also arrays of characters terminated by the null character (‘\0’).

Advantages

- Pointers also act as references to different types of objects such as variables, arrays, functions, structures, etc. In C, we use pointer as a reference.
- Storage of strings through pointers saves memory space.
- Pointers may be used to pass on arrays, strings, functions, and variables as arguments of a function.
- Passing on arrays by pointers saves lot of memory because we are passing on only the address of array instead of all the elements of an array, which would mean passing on copies of all the elements and thus taking lot of memory space.

Declaring Pointer Variables

Syntax

```
data_type *pt_name;
```

Here,

- The `*` tells that the variable pt_name is a name of the pointer variable.
- `Pt_name` needs a memory location.
- `Pt_name` points to a variable of type data_type.

Declaring Pointer Variables

Example

```
int *p;
char *name;
float *num;
```

Initiaization of Pointer Variables

- The process of assigning the address of a variable to a pointer variable is known as initialization.
- All uninitialized pointers will have some unknown values that will be interpreted as memory addresses.
- They may not be valid addresses or they may point to some values that are wrong.
- Once a pointer variable has been declared we can use the assignment operator to initialize the variable.

Initiaization of Pointer Variables

Example

```
int a;
int *p;
p = &a;
```

```
int a;
int *p = &a;
```

```
int a, *p - &a;
```

```
int *p = &a, a; \\Invalid
```

Initialization of Pointer Variables

- We can also define a pointer variable with an initial value of `NULL` or `0`;

```
int *p = null;
int *q = 0;
```

Accessing Variables through Pointers

```c
#include <stdio.h>
int main(void){
    //normal variable
    int num = 100;
    //pointer variable
    int *ptr;
    //pointer initialization
    ptr = &num;
    //pritning the value
    printf("value of num = %d\n", *ptr);
    return 0;
}
```

Another Example

```c
#include <stdio.h>
void main()
{
    int x,y;
    int *ptr;
    x=10;      ptr=&x;    y=*ptr;
    printf("Value of x is %d\n",x);
    printf("%d  is stored  at address %u\n",x,&x);
    printf("%d is stored at address %u\n",*&x, &x);
    printf("%d is stored at address %u\n",*ptr,ptr);
    printf("%d is stored at address %u\n",ptr,&ptr);
    printf("%d is stored at address %u\n",y,&y);
    *ptr=100;
    printf("\nNew value of x =%d\n",x);
}
```

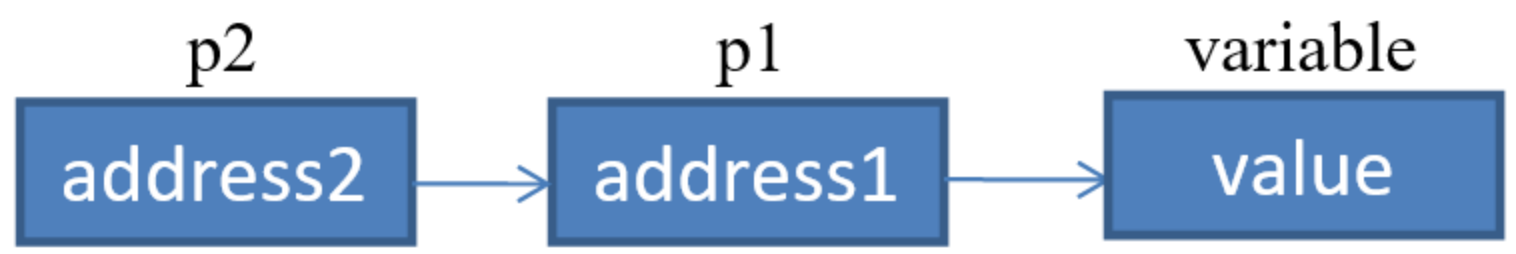
Output

```
Value of x is 10
10  is stored  at address 6422092
10 is stored at address 6422092
10 is stored at address 6422092
6422092 is stored at address 6422080
10 is stored at address 6422088

New value of x =100
```

Chain of Pointers

- We can create a chain of pointers by Pointing a pointer to point another pointer.



- The pointer variable `p2` contains the address of the pointer variable `p1`, which points to the location that contains the desired value.
- This is known as multiple indirections.

Chain of Pointers

- A variable that is pointer to a pointer must be declared using additional indirection operator symbol in front of the name.

```
int **p2;
```

- The pointer `p2` is not a pointer to an integer, but rather a pointer to an integer pointer.
- We can access the target value indirectly pointed to by pointer to a pointer by applying the indirection operator twice.

Chain of Pointers

```c
#include <stdio.h>
void main(){
    int x, *p1,**p2; x=100;
    p1=&x; p2=&p1;
    printf("pointer to pointer value %d",**p2);
}
```

Output

```
pointer to pointer value 100
```

Pointer Expressions

Pointer variables can be used in expressions.

```
y = *p1 * *p2; //y = (*p1) * (*p2)
*p2= *p2 + 10; p1+4;
p2-2;
p1-p2;
p1++;
-p2;
sum += *p2;
```

Pointer Expressions

In addition to the arthematic operations we can use relational operators also

```
p1>p2
p1==p2
p1 != p2
```

But below are invalid

```
p1/p2
p1 * p2
p1/3
```

Pointer Increment & Scale Factor

- If `p1` is a pointer then `p1++` point to the next value of its type.
- If `p1` is an integer pointer with an initial value, say `4020`, then the operation `p1++`, the value of `p1` will be `4022`.
- i.e., the value increased by the length of the data type that it points to

```
char      1 byte
int       2 bytes
float     4 bytes
long int  4 bytes
double    8 bytes
```