7. Pointers

Basics of pointers, pointer to pointer , pointer and array , pointer to array, array of pointers, functions returning a pointer

Total Hours: 04

Module Weightage: 9%

Table of contents

Introduction- Basics of Pointers

- Pointer is a variable that stores/points the address of another variable.
- It is used to allocate memory dynamically i.e. at run time.
- The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

Introduction

- Pointer Syntax : data_type *var_name;
  Example : int *p; char *p;
- Where, * is used to denote that “p” is pointer variable and not a normal variable.

Key points about pointers

- Normal variable stores the value whereas pointer variable stores the address of the variable.
- The content of the C pointer always be a whole number i.e. address.
- Always C pointer is initialized to null, i.e. int *p = null.
- The value of null pointer is 0.

Key points about pointers

- & symbol is used to get the address of the variable.
- * symbol is used to get the value of the variable that the pointer is pointing to.
- If a pointer in C is assigned to NULL, it means it is pointing to nothing.
- Two pointers can be subtracted to know how many elements are available between these two pointers.

Key points about pointers

- But, Pointer addition, multiplication, division are not allowed.
- The size of any pointer is 2 byte (for 16 bit compiler).

Print Pointer

- To print the address of a variable, we use "%p" specifier in C language. There are two ways to get the address of the variable:
- By using "address of" (&) operator
- By using pointer variable

using "Address of" (&) operator

```
#include <stdio.h>
int main(void){
    // declare variables
    int a; float b;  char c;
    printf("Address of a: %p\n", &a);
    printf("Address of b: %p\n", &b);
    printf("Address of c: %p\n", &c);
    return 0; }
```

```
Address of a:
0x7ffff3946390
Address of b:
0x7ffff3946394
Address of c:
0x7ffff394638f
```

Using Pointer Variable

```
#include <stdio.h>
int main(){
    int *ptr, a=5;
    ptr=&a;
    printf("%d,%d",*ptr,a);
    printf("\n%p",ptr);
    return 0;
}
```

```
/* Output:
 * 5,5
 *
0x7ffdd15c862c
 */
```

Value of a variable incremented

```
#include <stdio.h>
int main(){
    int *ptr, a=5;
    ptr=&a;
    *ptr+=2;//incrementing value of by 2
    printf("%d,%d",*ptr,a);
    return 0; }
```

Value of a variable incremented

```
/*
Output:
7,7
*/
```

Pointer to Pointer

- A pointer to a pointer is a form of multiple indirection, or a chain of pointers.
- Normally, a pointer contains the address of a variable. When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.

Pointer to Pointer

- Pointer to Pointer in C
- A variable that is a pointer to a pointer must be declared as such. This is done by placing an additional asterisk in front of its name. For example, the following declaration declares a pointer to a pointer of type int –
- int **var;

Pointer to Pointer

```
#include <stdio.h>
 int main () {
   int  var;    int  *ptr;    int  **pptr;
   var = 3000;
   /* take the address of var */
   ptr = &var;
```

Pointer to Pointer

```
  /* take the address of ptr using address of
operator & */
  pptr = &ptr;
  /* take the value using pptr */
  printf("Value of var = %d\n", var );
  printf("Value available at *ptr = %d\n", *ptr
);
  printf("Value available at **pptr = %d\n",
**pptr);
```

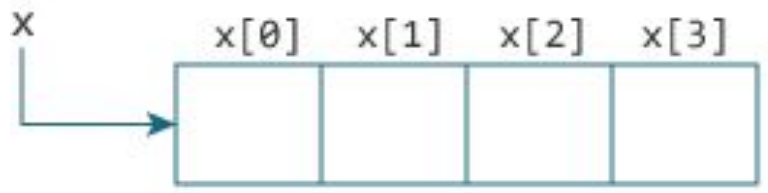
Pointer to Pointer

```
  /* take the address of ptr using address of
operator & */
  pptr = &ptr;
  /* take the value using pptr */
  printf("Value of var = %d\n", var );
  printf("Value available at *ptr = %d\n", *ptr
);
  printf("Value available at **pptr = %d\n",
**pptr);
```

Pointer to Pointer

```
/*
Output:
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
*/
```

Pointer and Array

- 

- &x[1] is equivalent to x+1 and x[1] is equivalent to *(x+1).
- &x[2] is equivalent to x+2 and x[2] is equivalent to *(x+2).
- ...
- Basically, &x[i] is equivalent to x+i and x[i] is equivalent to *(x+i).

Pointer and Array 1-D Example

```
#include<stdio.h>
int main(){
short
arr[13]={2,7,10,12,15,18,25,32,35};
printf("%d,%d",*(arr),*(arr+1));
return 0;
}
```

```
/*
 Output:
2,7
 */
```

Pointer and Array 2-D Example

```
#include<stdio.h>
int main(){
short
arr[3][3]={2,7,10,12,15,18,25,32,35};
printf("%d,%d",*(arr+1)[1],**(arr+1));
return 0; }
```

```
/* Output:
25,12
 */
```

Pointer and Array Example

```
#include<stdio.h>
int main(){
char *s="CKPCET";
char *p=s;
printf("%c%c",*(p+1),s[1]);
return 0;
}
```

```
/*
 Output:
 KK
 */
```

Pointer to Array Example

```
#include<stdio.h>
int main(){
        int arr[5] = { 1, 2, 3, 4, 5 };
        int *ptr = arr;
        printf("%p\n", ptr);
        printf("%d\n", *ptr);
        return 0;}
```

```
/*
Output:
0x7ffee51c4aa0
1
*/
```

a pointer ptr that points to the 0th element of the array.

Passing Pointer to Function

```
#include <stdio.h>
int main(void){
void fun(int *a, int *b);
int x=5, y=7;
printf("Before calling fun: %d,%d\n",x,y);
fun(&x,&y);
printf("After calling fun: %d,%d",x,y);
}
```

Passing Pointer to Function

```
void fun(int *a, int *b){
*a=++*a;
*b=++*b;
}
/* Output:
Before calling fun: 5,7
After calling fun: 6,8*/
```

Function returning Pointer

```
#include <stdio.h>
int main(void){
int *incr(int* x);
int x=5;
int *y=incr(&x);
printf("%d",*y);
}
```

```
int *incr(int* y){
++*y;
return(y);
}


/*
 * Output: 6*/
```

Array of Pointers

```
#include <stdio.h>
 const int MAX = 3;
 int main () {
   int  var[] = {10, 100, 200};
   int i, *ptr[MAX];
   for ( i = 0; i < MAX; i++) {
      ptr[i] = &var[i]; /* assign the address of
integer. */    }
```

Array of Pointers

```
  for ( i = 0; i < MAX; i++) {
    printf("Value of var[%d] = %d\n", i, *ptr[i]
);
  }

  return 0;
}
```

Array of Pointers

Output:

```
Value of var[0] = 10
Value of var[1] = 100
Value of var[2] = 200
```

Pointer example

```
#include <stdio.h>
int main(){
    int num[3][2]={2,7,10,12,15,18};
    printf("%d %d",*(num+1)[1],**(num+2));
    return 0;
}
/*Output:
15,15*/
```

Good Example

```
#include <stdio.h>
int main(void) {
    int i;
    int *ptr=(int *)malloc(5*sizeof(int));
    for(i=0;i<5;i++)
        *(ptr+i)=i;
```

Good Example

```
printf("%d",*ptr++);
    printf("%d",(*ptr)++);
    printf("%d",*ptr);
    printf("%d",*++ptr);
    printf("%d",++*ptr);
    return 0;
}
/*Output: 0 1 2 2 3*/
```

References

1. Pointers in C, https://fresh2refresh.com/c-programming/c-pointer/
2. Pointer Introduction, https://www.includehelp.com/c-programs/printing-an-address-of-a-variable.aspx
3. Contents here are also taken from the Assignments I solved while attending NPTEL course titled “Problem solving through Programming In C”, 2019.
4. Pointer and Array, https://www.programiz.com/c-programming/c-pointers-arrays
5. Pointer to Array, https://www.studytonight.com/c/pointers-with-array.php
6. Array of pointers, https://www.tutorialspoint.com/cprogramming/c_array_of_pointers.htm