International Journal of Engineering, Management & Technology (IJEMT)
www.ijemt.com, Volume 1 Issue X, December 2022, PP 38-41, ISSN (Online): 2583 – 4517

An Analysis of Pointers in C Language: Address Calculation

Pankaj Kumar Gupta[1*], Mayank Sharma[2]
1.Assistant Professor & Head, BCA Department, DPBS (PG) College, Anupshahr Distt. Bulandshahr (UP) India.
2.Assistant Professor, BCA Department, DPBS (PG) College, Anupshahr Distt. Bulandshahr (UP) India.

ABSTRACT: *There are various tasks inside the C language that become very easy when it is done by Pointers, whereas if we look at dynamic memory allocation(DMA), then it can be possible only and only by Pointer, so to become a good programmer, it is very important to learn the concept of Pointers, by any person, so now let's talk, what is a pointer? Answer is: Pointer is a special type of variable that can contain the address of another variable. In this research paper we will discuss about the calculation of address using a pointer. Here pointers in C language is discussed for same purpose.*

KEYWORDS: *Pointer, DMA, Indirection, Address-of, Array.*

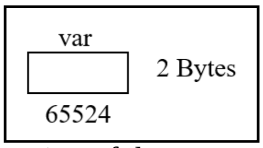
INTRODUCTION

There are various tasks inside the C language that become very easy when it is done by Pointers, whereas if we look at dynamic memory allocation(DMA), then it can be possible only and only by Pointer, so to become a good programmer, it is very important to learn the concept of Pointers, by any person, so now let's talk, what is a pointer? Answer is: Pointer is a special type of variable that can contain the address of another variable.

Let's have an example:

int var;

This is the declaration of a variable named var. This declaration tells to the compiler that create a space of 2 bytes in the memory and label it as var.



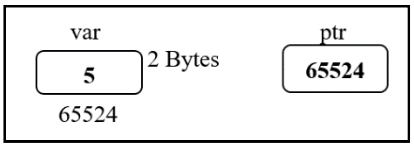
Anything if declared, compiler creates spaces for it somewhere in the memory. The location of that space is called as address. Here 65524 is the address of var. This address is stored in a special variable which is known as pointer.

Declaring a pointer:

To declare a pointer variable, we use an asterisk (*) symbol. an asterisk (*) symbol is succeeded by a pointer variable name.

Like: int *ptr; *//this pointer variable (*ptr) is now able to keep the address of other variable (int) var.*

int var, *ptr;
var = 5;
ptr = &var; *//here ampersand symbol (&) is used to*
// represent the address of operator.



Pointer Operators:

There are two pointer operators that are used in C language. Both operators fall in the category of Unary operators (i.e. they can be used on single operands only).

* **Manuscript Received on: Sep 12, 2022**
Manuscript Published on www.ijemt.com : Dec 28, 2022
Corresponding Author: Pankaj Gupta

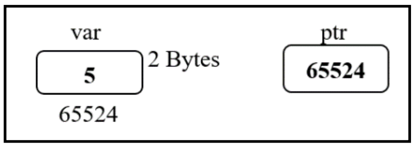
1. **Value-at Operator:** it is a pointer operator that tells the value at any pointer (Address). It is also known as indirection operator. It is used to dereference a pointer (Address). So it's another name is dereference pointer operator. It is represented by a special symbol asterisk (*).
   int x;
   x = *ptr; //*Here x variable is assigned with value at pointer variable ptr;*
   //now x will contain 5.
2. **Address-of Operator:** it is a pointer operator that tells the address of a value (variable), where it is assigned. It is represented by a special symbol ampersand (&).
   int *ptr;
   ptr = &var; //*Here ptr variable is assigned with address of variable var;*
   //now ptr will contain the address of var (i.e. 65524 according to above example).

TYPES OF POINTERS

Pointers are categorized into various categories. Some of those are as follows:
1. Pointer to Integer: It is a variable, that stores the address of any integer variable only.
For Example: int *ptrvar; //here ptrvar is a pointer to integer
2. Pointer to Float: It is a variable, that stores the address of any float variable only.
For Example: float *ptrvar; //here ptrvar is a pointer to float
3. Pointer to Character: It is a variable, that stores the address of any character variable only.
For Example: char *ptrvar; //here ptrvar is a pointer to character
4. Pointer to Double: It is a variable, that stores the address of any double variable only.
For Example: double *ptrvar; //here ptrvar is a pointer to double
5. Pointer to Long: It is a variable, that stores the address of any long variable only.
For Example: long *ptrvar; //here ptrvar is a pointer to long
6. Pointer to Pointer: It is a variable, that stores the address of any other pointer (ie. address) variable only.
For Example: int **ptrvar; //here ptrvar is a pointer to pointer (only for integer)
7. Pointer to Void: It is a variable, that stores the address of any type of variable only.
For Example: void *ptrvar; //here ptrvar is a pointer to void

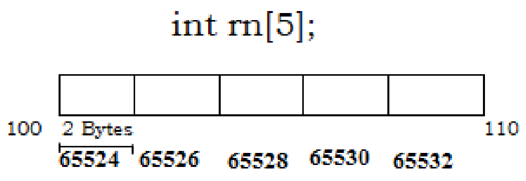
Pointer Arithmetic/Address Calculation:
We use the concept of Pointer Arithmetic for finding the address of some variable using the address(through pointer) of some other variable. For same purpose we require the Calculation of Addresses just like mathematics. Here all the operations of Arithmetic except Multiply/Divide/Modulo Division work in same way.
Operators that work for this purpose are addition (+), subtraction (-), pre/post increment (++), pre/post decrement (--).
*Operators that do not work: multiplication (*), division (/), modulo division (%).*
Let us consider following example:

```
int var, *ptr, *nextptr;
var = 5;
ptr = &var;            //ptr = 65524
nextptr = ptr + 1;     //nextptr = 65526
```
since an integer takes 2 bytes to store an integer value.
```
nextptr = ptr - 1;     //nextptr = 65522
nextptr = ++ptr;       //nextptr = 65524 and ptr = 65524
nextptr = --ptr;       //nextptr = 65522 and ptr = 65522
```



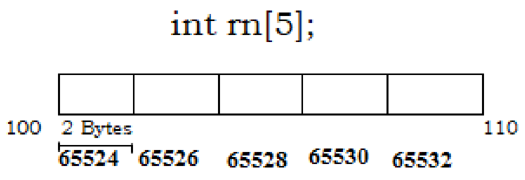
Address calculation in Arrays:
A collection of homogenous data elements is known as Array. A C*ommon name is shared by all the elements and same are stored on contiguous locations.*

For example:
If we want to store the Roll Numbers of 5 Students(in integer form), it will be termed as array of roll numbers/integers.
int rn[5];   //roll numbers of 5 students
Memory Allocation for above array is:
//since integer takes 2 bytes in the memory, so increment in locations of every is done by 2.



All the elements are stored here on the contiguous addresses as shown in above figure. If we know the address of first element, then we can find the addresses of other elements through the concept of calculations.
Like:
int *ptr,*nextptr;
ptr = &rn[0];   //ptr = 65524
nextptr = ptr + 1;   //nextptr = 65526 that means the location of 2nd student's *roll number*.
nextptr = ptr + 2;   //nextptr = 65528 that means the location of 3rd student's *roll number.*
nextptr = ptr + 3;   //nextptr = 65530 that means the location of 4th student's *roll number.*
nextptr = ptr + 4;   //nextptr = 65532 that means the location of 5th student's *roll number.*

Array versus Pointer:
The name of an Array is a pointer constant that stores the address of starting element of array.
For Single Dimensional Array:
For example:
int rn[5], *ptr;
ptr = &rn[0];   //ptr = 65524
This statement can be written as
ptr = rn;   //ptr = 65524
ptr++;   //now ptr = 65526 that means the location of 2nd student's *roll number.*
ptr = ptr + 2;   //now ptr = 65530 that means the location of 4th student's *roll number.*
ptr++;   //now ptr = 65532 that means the location of 5th student's *roll number.*



Accessing the elements through pointer/name of Array:

*(rn + 0) or *rn or rn[0] or *(rn)	*//Roll number of 1st Student.*
rn + 0 or rn or &rn[0]	*//Location for Roll number of 1st Student.*
*(rn + 1) or rn[1]	*//Roll number of 2nd Student.*
rn + 1 or &rn[1]	*//Location for Roll number of 2nd Student.*
*(rn + 2) or rn[2]	*//Roll number of 3rd Student.*
rn + 2 or &rn[2]	*//Location for Roll number of 3rd Student.*
*(rn + 3) or rn[2]	*//Roll number of 4th Student.*
rn + 3 or &rn[3]	*//Location for Roll number of 4th Student.*
*(rn + 4) or rn[2]	*//Roll number of 5thStudent.*
rn + 4 or &rn[4]	*//Location for Roll number of 5th Student.*

For Double Dimensional Array:
For example:
int m[2][3] = {{11,22,33},{44,55,66}}; //*An arr*ay to store the marks of 2 students in 3-3 subjects.

Memory Allocation is considered to be Row Major Form.
Array variable names   Addresses

m[0][0]	m[0][1]	m[0][2]
m[1][0]	m[1][1]	m[1][2]

65524	65526	65528
65530	65532	65534

Elements (Marks)

11	22	33
44	55	66

int *ptr;
ptr=&m[0][0];
To access all elements through pointer:
m[0][0] or *(m[0]+0) or *(*(m+0)+0) or *(*(m+0)) or **m or *(ptr[0]+0) or *(*(ptr+0)+0) or *(*(ptr+0)) or **ptr *//shows the marks of first student in first subject.*

m[0][1] or *(m[0]+1) or *(*(m+0)+1) or *(*m+1)) or *(ptr[0]+1) or *(*(ptr+0)+1) or *(*ptr+1)
//shows the marks of first student in second subject.
m[0][2] or *(m[0]+2) or *(*(m+0)+2) or *(*m+2)) or *(ptr[0]+2) or *(*(ptr+0)+2) or *(*ptr+2)
//shows the marks of first student in third subject.
m[1][0] or *(m[1]+0) or *(*(m+1)+0) or *(*(m+1)) or *(ptr[1]+0) or *(*(ptr+1)+0) or *(*(ptr+1))
//shows the marks of second student in first subject.
m[1][1] or *(m[1]+1) or *(*(m+1)+1) or *(ptr[1]+1) or *(*(ptr+1)+1)
//shows the marks of second student in second subject.
m[1][2] or *(m[1]+2) or *(*(m+1)+2) or *(ptr[1]+2) or *(*(ptr+1)+2)
//shows the marks of second student in third subject.
This was all about the address calculation of any desired variable through pointers.

References

1. www.biitonline.co.in a website for learning Engineering subjects for Computer Engineers designed & written by **Pankaj Kumar Gupta (First Author of this paper)**.
2. International Journal of Essential Sciences Vol.14 No.1&2 2020. Topic is “Arrays: A Theoretical Approach of Memory Allocation By Pankaj Kumar Gupta and Dr. P. K. Tyagi”.
3. International Journal of Research in all subjects multi languages vol: 10 Issue: 3 March 2022 ISSN 2321-2853. Topic is “Conceptual Discussion on Operations of Array: Traversal, Insertion& Deletion” By Pankaj Kumar Gupta and Dr. P. K. Tyagi”.
4. *International Journal of Engineering, Management & Technology (IJEMT) www.ijemt.com, Volume 1 Issue II, April 2022, PP 49-55, ISSN (Online): 2583 – 4517. Topic is “* Conceptual & Mathematical Analysis on Operations of Array: Searching” By Pankaj Kumar Gupta, Prof. R. K. Agrawal.
5. Computer Science with C++ By SumitaAroraby SumitaArora.
6. Let Us C by YashavantKanetkar
7. Sartajsahni, “Data structures, algorithms and applications in C++”, University press.
8. Seymour Lipschutz. “Theory and problems of data structures”, Tata Mcgraw hill international editions”.
9. The C Programming Language by Brian W. Kernighan / Dennis Ritchie.
10. www.en.wikipedia.org/wiki/Array.
11. www.geeksforgeeks.org
12. www.w3schools.com