KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**

****

Seminar Report

**TITLE**

**“POINTERS IN C”**

*Submitted in the partial fulfilment for the academic requirement of*

**1st Semester B.E in**

**PROBLEM SOLVING WITH C**

Submitted by

|  |  |
| --- | --- |
| **NAME OF THE CANDIDATE** | **ROLL NO** |
| **SHRADHA MALLIKARJUN PATIL** | **I 02** |
| **BASAVRAJ DUGGANAVAR** | **I 08** |
| **OMPRAKASH JAT** | **I 47** |
| **SMRUTI BHANDIWAD** | **I 49** |

**GUIDED BY: R. S. Patil Sir**

**Course Seminar report and ppt content**

**Marks’s allocation:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Batch No.: 08 | | | | | |
| 1. | Seminar Title: | Marks Range | USN | | | |
|  |  |  |  |
| 2. | Abstract (PO2) | 0-2 |  |  |  |  |
| 3. | Application of the topic to the course (PO2) | 0-3 |  |  |  |  |
| 4. | Literature survey and its findings (PO2) | 0-4 |  |  |  |  |
| 5. | Methodology, Results and Conclusion (PO1, PO3,PO4) | 0-6 |  |  |  |  |
| 6. | Report and Oral presentation skill (PO9, PO10) | 0-5 |  |  |  |  |
|  | Total | 20 |  |  |  |  |

**TABLE OF CONTENTS:**

1. ABSTRACT
2. INTRODUCTION
3. DECLARATION AND INITIALIZATION OF POINTERS
4. TYPES OF POINTERS
5. POINTER ARITHMATIC
6. POINTERS AND ARRAYS
7. POINTERS AND STRINGS
8. ADVANTAGES AND DISADVANTAGES OF POINTERS
9. CONCLUSION
10. REFERENCES

**ABSTRACT:**

# This is a seminar report made on the topic ‘Pointers in C’. A pointer is a derived data type in C. Pointers contains memory addresses as their values. In this we have explained about the basics of pointers that is declaration and initialization of pointers, types of pointers , pointer arithmetic, pointers and arrays, pointers and strings, advantages and disadvantages of pointers.

INTRODUCTION:

The pointer in C language is a variable which stores the address of another variable. This variable can be of type int, char, array, function, or any other pointer. The size of the pointer depends on the architecture.  C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers. . A pointer *references* a location in memory, and obtaining the value stored at that location is known as [*dereferencing*](https://en.wikipedia.org/wiki/Dereference_operator) the pointer.

DECLARATION AND INITIALIZATION OF POINTERS:

Like variables, pointers in C programming have to be declared before they can be used in your program. Pointers can be named anything you want as long as they obey C's naming rules. A pointer declaration has the following form.

data\_type \* pt\_name;

Here,

The asterisk(\*) tells that the variable pt\_name is a pointer variable.

data\_typeis the pointer's base type of C's variable types and indicates the type of the variable that the pointer points to.

The process of assigning the address of a variable to a pointer variable is known as initialization. Once a pointer variable has been declared we can use the assignment operator to initialize the pointer variable.

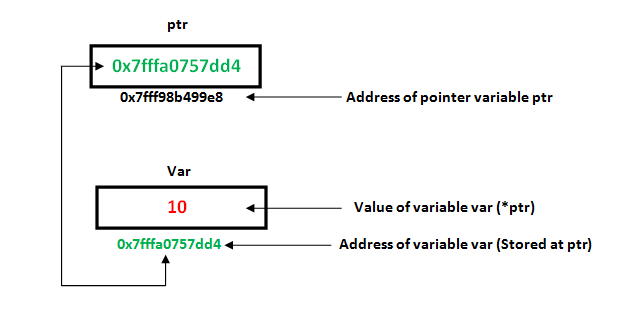
For example, consider the code snippet below

int \*p, a;

p=&a;

Here the address of the integer variable ‘a’ is stored in pointer variable ‘p’.

The ampersand operator (&) gives the address of the variable and \*p gives the value stored at the address in the pointer variable.



C Program to demonstrate declaration and initialization of pointers:

#include <stdio.h>

int main () {

int var = 20; /\* actual variable declaration \*/

int \*ip; /\* pointer variable declaration \*/

ip = &var; /\* store address of var in pointer variable\*/

printf("Address of var variable: %x\n", &var );

/\* address stored in pointer variable \*/

printf("Address stored in ip variable: %x\n", ip );

/\* access the value using the pointer \*/

printf("Value of \*ip variable: %d\n", \*ip );

return 0;

}

When the above code is compiled and executed, it produces the following result −

Address of var variable: bffd8b3c

Address stored in ip variable: bffd8b3c

Value of \*ip variable: 20

TYPES OF POINTERS:

* Null Pointer:

We can create a null pointer by assigning null value during the pointer declaration. This method is useful when you do not have any address assigned to the pointer. A null pointer always contains value 0.

* Void Pointer:

A void pointer is also called as a generic pointer. It does not have any standard data type. A void pointer is created by using the keyword void. It can be used to store an address of any variable.

* Wild pointer:

A pointer is said to be a wild pointer if it is not being initialized to anything. These types of C pointers are not efficient because they may point to some unknown memory location which may cause problems in our program and it may lead to crashing of the program. One should always be careful while working with wild pointers.

Other types of pointers also include,

* Dangling pointer
* Complex pointer
* Near pointer
* Far pointer
* Huge pointer

POINTER ARITHMATIC:

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 operations that can be used on pointers:

1. Increment/Decrement of a Pointer
2. Addition of integer to a pointer
3. Subtraction of integer to a pointer
4. Subtracting two pointers of the same type

**Increment:** It is a condition that also comes under addition. When a pointer is incremented, it actually increments by the number equal to the size of the data type for which it is a pointer.  
**For Example:**  
If an integer pointer that stores **address 1000** is incremented, then it will increment by 2(**size of an int**) and the new address it will points to **1002**. While if a float type pointer is incremented then it will increment by 4(**size of a float**) and the new address will be **1004**.

**Decrement:** It is a condition that also comes under subtraction. When a pointer is decremented, it actually decrements by the number equal to the size of the data type for which it is a pointer.  
**For Example:**  
If an integer pointer that stores **address 1000** is decremented, then it will decrement by 2(**size of an int**) and the new address it will points to **998**. While if a float type pointer is decremented then it will decrement by 4(**size of a float**) and the new address will be **996**.

**Addition:** When a pointer is added with a value, the value is first multiplied by the size of data type and then added to the pointer.

**Subtraction**: When a pointer is subtracted with a value, the value is first multiplied by the size of the data type and then subtracted from the pointer.

C Program to demonstrate pointer arithmetic:

#include <stdio.h>

int main()

{

int N = 4;

int \*ptr1, \*ptr2;

ptr1 = &N;

ptr2 = &N;

printf("Pointer ptr1 before Increment: %d \n", ptr1);

ptr1++;

printf("Pointer ptr1 after Increment: %d \n", ptr1);

printf("Pointer ptr1 before Decrement: %d \n", ptr1);

ptr1--;

printf("Pointer ptr1 after Decrement: %d \n", ptr1);

return 0;

}