


# Pointers

# Unit 9

## INTRODUCTION

In the previous unit 8 we have studied about C structure and their declarations, definitions, initializations. Also we have taught importance of C structures and their applications. In this unit we will describe about another important feature of C language like pointer. In this unit, we will describe in detail how a pointer is defined, initialized; how pointer works in a program and we will know the benefits of uses of pointer.

<p>Timeframe</p>  <p>How long ?</p>	<p>We expect that this unit will take maximum 6 hours to complete.</p>
--	--

<p><b>Unit Structure</b></p>	
<p>Lesson- 1 : Introduction to C Pointers</p> <p>Lesson- 2 : Pointer Declaration and Initialization</p> <p>Lesson- 3 : Accessing Variable and Pointer Expression</p>	

**Lesson-1****Introduction to C Pointers****Learning Outcomes****Outcomes**

Upon completion of this lesson you will be able to

- Understand basic idea about C Pointers.
- Learn necessity of pointers in C program.
- Understand the advantages and disadvantages of pointers.

	<b>Keywords</b>	<b>Pointer, Advantages, Disadvantages, Address</b>
---	-----------------	--

**PRELIMINARY CONCEPT OF POINTERS**

Pointers are the powerful feature of C language. It is easy and fun to learn. Actually, pointers are used in C program to access the memory and manipulate the memory address. A number of C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation. Dynamic memory allocation cannot be performed without using pointers. So it becomes necessary to learn pointers to become a perfect C programmer.

**Benefits of pointers in c**

The usages of pointers will give the following benefits:

1. By using pointers we can access a data which is available outside a function.
2. By using pointer variable we can implement dynamic memory allocation.
3. Pointers provide a way to return more than one value to the functions.
4. Pointer reduces the storage space and complexity of the program.
5. Pointer reduces the execution time of the program and increases the execution speed of program.
6. Provides an alternate way to access array elements.
7. Pointers can be used to pass information back and forth between the calling function and called function.
8. A pointer allows us to perform dynamic memory allocation and de-allocation.
9. With the help of pointer we can build complex data structures such as linked list, stack, queues, trees, graphs etc.
10. A pointer allows us to resize the dynamically allocated memory block.

**Disadvantages of pointers in C**

The drawbacks of pointers in c are mentioned bellow:

1. Uninitialized pointers might cause segmentation fault.

2. Dynamically allocated block needs to be freed explicitly. Otherwise, it would lead to memory leak.
3. Pointers are slower than normal variables.
4. If pointers are updated with incorrect values, it might lead to memory corruption.

**WHAT IS POINTER?**

A pointer is a nothing but a variable that contains an address of another same type of variable. Since, a pointer is a variable; its value is stored in the memory in another location. Suppose, we have two variables such as *value1* and *value2* and we want to assign the address of *value1* to variable *value2*. The link between the variables *value1* and *value2* can be visualized as shown in figure 9.1.1.

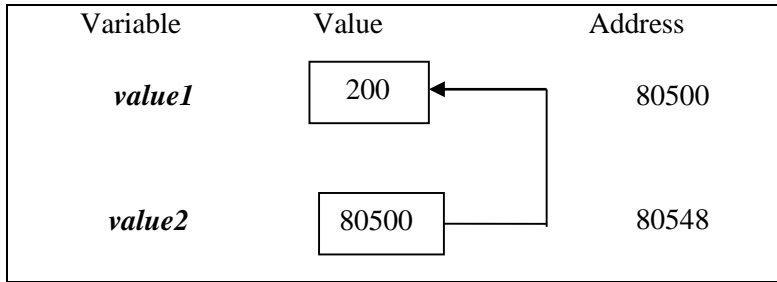


Figure 9.1.1: Pointer variable concept

Here, the address of *value2* is 80548. Since, the value of the variable *value2* is the address of the variable *value1*, we may access the value of *value1* by using the value of *value2* and therefore, we say that the variable *value2* points to the variable *value1*. Therefore, *value2* is said to be pointer that points the *value1*.



Note it!

---

*A pointer is a nothing but a variable that contains an address of another same type of variable.*

---

**ACCESSING THE ADDRESS OF A VARIABLE**

In C language, the address of a variable can be known with the help of the operator ‘&’. The operator ‘&’ immediately prior a variable returns the address of the variable associated with it. For instance, the statement: `value2=&value1;` would assign the address 80500(the location of variable *value1* in figure 7.1.1 to the variable *value2*. The ‘&’ operator can be said as ‘address of’. This operator can be used only with a simple variable or an array element.



Activity


1. Mention the significance of using pointers in C language by your own idea.  
 .....  
 .....



Study skills

1. Mention the significance of the following statement:

`value2=&value1;`

<p><b>Summary</b></p>  <p>Summary</p>	
<p><b>In this lesson we have</b></p> <ul style="list-style-type: none"> <li>▪ Learned about basic idea about C pointers.</li> <li>▪ Learned advantages and disadvantages of pointer.</li> <li>▪ Also understood how variables are accessed by pointer in program.</li> </ul>	

**ASSIGNMENT**



Assignment

1. Declare three integer type variables and two integer type pointers and show how the variables are accessed by declared pointers.  
.....

**Assessment**



Assessment

**Multiple Choice Questions (MCQ)**

1. A pointer is a variable that contains an address of-
  - a) Another same type of variable
  - b) Another different type of variable
  - c) Another function
  - d) None of these
2. Dynamic memory allocation cannot be performed without-
  - a) Using pointers
  - b) Using function
  - c) Using variable
  - d) Using array
3. Pointer reduces the
  - a) Programming speed
  - b) Execution time of the program
  - c) Memory size
  - d) None of these
4. Pointer increases the
  - a) Execution speed of program.
  - b) Memory Size
  - c) Execution time of program
  - d) None of these
5. The address of a variable can be known with the help-
  - a. Of the operator '&'
  - b. Of the operator '#'
  - c. Of the operator '\$'
  - d. Of the operator '@'

**Exercises**

1. What is pointer?
2. Explain the advantages and disadvantages of pointer
3. How pointers access the address of a variable? Explain.

## Lesson-2

## Pointer Declaration and Initialization

### Learning Outcomes



### Outcomes

Upon completion of this lesson you will be able to

- Understand declaration procedure of pointer.
- Learn initialization procedure of pointer.

	<b>Keywords</b>	<b>Pointer, Declaration, Initialization</b>
--	-----------------	---

### DECLARING A POINTER

We know that a pointer is a variable that contains an address of another same type of variable. Since pointer variables contain addresses that belong to a separate data type, they must be declared as pointers before we use them. The general syntax or form of pointer declaration is as follows:

```
data-type * pointer_name;
```

Here, *data-type* is the valid C data type and *pointer\_name* is the pointer variable name. The above declaration tells the compiler three things about the variable *pointer\_name* :

- i. The asterisk (\*) tells that *pointer\_name* is a pointer variable
- ii. *pointer\_name* needs a memory location
- iii. *pointer\_name* points to a variable of type *data-type*

### Declaration Example

For example,

```
int *ptr;
```

declares the variable *ptr* as a pointer variable that points to an integer (i.e., int) data type. Likewise the following statement,

```
float *ptr;
```

declares *ptr* as a pointer to floating point variable. Once a pointer has been declared, it can be made to point to a variable using an assignment operator like:

```
int *ptr, value;
```

```
ptr = &value;
```

Where *ptr* to point variable *value* so pointer variable *ptr* contains the address of *value*. This process is also called pointer initialization. Before a pointer is initialized, it should not be used.

Such valid pointer declaration example as follows:

```
int a,b,*ptr;
```

```
float x,*y;
```

```
ptr = &a;
```

```
y = &x;
```



**Pointers are used in C program to access the memory and manipulate the memory address**

Such invalid pointer declaration example as follows:

```
int a,b,*ptr;
float x,*y;
ptr = &x; //Error, type mismatch, because pointer variable always points to same type of data.
y = &a; //Error, type mismatch, because pointer variable always points to same type of data.
```

**INITIALIZATION OF POINTER VARIABLE**

In general, pointer initialization is the process of assigning address of a variable to pointer variable. Pointer variable contains address of variable of same data type. In C language address operator ‘&’ is used to determine the address of a variable. In C, it is possible to initialize and declaration together.

For example:

```
int p = 100 ;
int *ptr ; //pointer declaration
ptr = &p ; //pointer initialization
or,
int *ptr = &p ; //initialization and declaration together
```

Here, *p* is integer variable and *ptr* is pointer variable. So initializes *ptr* to the address of *p*. Here note that this is an initialization of *ptr*, not *\*ptr* and also remember that the target variable *p* is declared first.



1. Mention the significance of using \* operator in pointer declaration.  
.....  
.....



Study skills

1. State the errors from the following code segment
 

```
void main()
{
    int ptr1, a,b;
    float *ptr, x, y;
    ptr1=&x;
    ptr=&b;
    x=&y;
    y=*ptr;
}
```

**Summary**



Summary

**In this lesson we have**

- Learned about C pointers definition and declaration.
- Learned initialization procedure of pointers.
- Also understood how pointers are used in program.
- Also understood pointer expression.

ASSIGNMENT



Assignment

- Write a program to determine the greatest common divisor (GCD) and least common multiple (LCM) of two integer number using structure.  
.....  
.....

Assessment



Assessment

Write ‘T’ for true and ‘F’ for false for the following statements:

- Pointer is a variable that contains an address of another same type of variable.
- The asterisk (\*) tells that *pointer\_name* is a pointer variable.
- Before a pointer is initialized, it should be used.

Multiple Choice Questions (MCQ)

- Pointers are used in C program to access-
  - The memory and manipulate the memory address
  - The variable
  - The functions
  - None of these
- Pointer variable contains address of variable of-
  - Different data type
  - Same data type
  - Different type function
  - Different type array
- Pointer initialization is the process of assigning address of a variable to
  - Variable
  - Function
  - Array
  - Pointer variable.

Exercises

- Explain pointer declaration procedure with example.
- Explain pointer initialization procedure with example.

**Lesson-3****Accessing Variable and Pointer Expression****Learning Outcomes****Outcomes**

Upon completion of this lesson you will be able to

- Explain how pointers are accessed variable.
- Explain pointer expression in a program.

	<b>Keywords</b>	<b>Pointer, Variable Access, Expression</b>
---	-----------------	---

**ACCESSING A VARIABLE THROUGH ITS POINTER**

When a pointer has been assigned the address of variable, in this situation the question is how to access the value of the variable using the pointer. Accessing a variable through its pointer we should follow three steps such as (a) at first we define a pointer variable (b) secondly, assign the address of a variable to a pointer and (c) finally access the value at the address available in the pointer variable. This is done by using indirection operator which is also known as unary operator (\*), that returns the value of the variable located at the address specified by its operand. Consider the following statements:

```
int data, *p, x;
data = 1000;
p=&data;
x=*p;
```

In the above statements, the 1<sup>st</sup> line declares *data* and *x* as integer variables and *p* as a pointer variable that pointing to an integer. The 2<sup>nd</sup> line assigns the value 1000 to variable *data* and then 3<sup>rd</sup> line assigns the address of *data* to the pointer variable *p* and then 4<sup>th</sup> line contains the indirection operator \*. When the operator \* is placed before a pointer variable in an expression, the pointer returns the value of the variable of which the pointer value is the address. In this situation, \* *p* returns the value of the variable *data*, because *p* is the address of *data*. For this reason, the value of *x* would be 1000. In the above statements, the two lines like

```
p=&data;
x=*p;
x=* &data;
x=data;
```

are equivalent to  
and also is equivalent to

---

**Program 9.3.1 Writes a program to illustrate use of indirection operator (\*) to access the value.**

---

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int a,b,*p;
a=200;
```



```

p=&a;
b=*p;
printf(" Value of a is:=%d\n",a);
printf(" The content of pointer p is:=%d \n",*p);
printf(" The content of variable b is:=%d\n",b);
getch();
}

```

**Output**

```

Value of a is:=200
The content of pointer p is:=200
The content of variable b is:=200

```

**POINTER EXPRESSIONS**

According to arithmetic operations or expression of other variables, pointer variables can be used in expressions. For instance, if *p1* and *p2* are declared as pointers then we can say that the following statements are valid:

```

A=*p1**p2;           this is same as (*p1) * (*p2)
SUM=SUM+*p1;
B=10*- *p2/ *p1;     this is same as (10*(-(*p2)))/(*p1)
*p2=*p2+100;

```

We can also add integers to or subtract integers from pointers as well as to subtract one pointer from another pointer. For instance,

```

p1 + 10;
p2 - 10;
p1- p2; are also valid statements.

```

We can also use short hand operators with the pointers such as

```

p1+ = 10;
sum + = *p2;
p1++; --p2; etc. are valid statements.

```

In C we can also compare pointers by using relational operators. The expressions such as

```

p1 > p2
p1 = p2 and
p1! = p2 are valid expressions.

```

But we cannot use pointers in division or multiplication. For instance, following expressions such as

```

p1/p2;
p1*p2;
p1/3; etc are not valid statements.

```

**Program 9.3.2 Write a program to illustrate the use of pointers in arithmetic operations.**

```

#include<stdio.h>#include<conio.h>
void main()
{
    int A,B,*p1,*p2,x,y,z;
    A=20;B=10;p1=&A;p2=&B;
    x=*p1 * *p2-5;
    y=((10* (-(*p2)))/(*p1))+30;
    printf("value of A = %d, and B = %d\n",A,B);
    printf("value of x = %d, and y= %d\n", x,y);
}

```

```
*p2 = *p2+10; *p1 =*p2-3;z = *p1 * *p2-5;
printf("value of A = %d,and B = %d\n", A,B);printf(" value of z = %d\n",z);
getch();
}
.....
Output
value of A = 20, and B = 10
value of x = 195, and y= 25
value of A = 17, and B = 20
value of z =335
```



Study skills

1. Describe what is wrong in the following pointer declaration:

```
void main(){
    int x,y,p1p2;p1=&x; p2=*p2;
}
```

2. Find the error(s) from the following code segments:

```
void main(){
    int *p1,*p2, x,y,z,mul,sum;
    x=p1-p2; y=p2/10; sum=sum+p2;mul=mul*p2;
    p2=&*x;z=*p1 + *p2++;
}
```

**Summary**



Summary

**In this lesson we have**

- Learned how variables are accessed by pointer.
- Also understood how pointers are used in program and understood pointer expression.

**ASSIGNMENT**



Assignment

1. Write a program to determine the summation of three integers values using pointers

.....  
 .....

**Assessment**



Assessment

**Write ‘T’ for true and ‘F’ for false for the following statements**

1. Accessing a variable through the pointer we should follow three steps.
2. Like other variables, pointer variables can be used in expressions.
3. We cannot use pointers in division or multiplication.

**Exercises**

1. Explain how variables are accessed using pointer.
2. Write a program that determines summation, subtraction, multiplication and division using pointer variable.

**UNITS TRUE / FALSE ANSWER**

Unit 4 :	Lesson 2 :	1→T, 2 →F, 3→T
	Lesson 3 :	1→T, 2 →T
	Lesson 4 :	1→T, 2 →F
Unit 6 :	Lesson 5 :	1→F, 2→T, 3→T, 4→T
Unit 7 :	Lesson 1 :	1→T, 2→T, 3→T, 4→F
	Lesson 7 :	1→T, 2→T, 4→F
	Lesson 8 :	1→F, 2→T, 3→T, 4→F, 5→F, 6→T, 7→T
	Lesson 9 :	1→F, 2→F, 3→F, 4→T, 5→T
Unit 9 :	Lesson 2 :	1→T, 2→T, 3→F
	Lesson 3 :	1→T, 2→T, 3→T

**UNITS MCQ ANSWER**

Unit 1 :	Lesson 4 :	1→ a, 2→ b, 3→ a, 4→ b, 5→ b
Unit 2 :	Lesson 4 :	1→ d, 2→ c
Unit 3 :	Lesson 2 :	1→ c, 2→ a, 3→ a
Unit 4 :	Lesson 1 :	1→ c, 2→ a
	Lesson 2 :	1→ a, 2→ a
Unit 6 :	Lesson 5 :	1→ a, 2→ a, 3→ c, 4→ b
Unit 7 :	Lesson 1 :	1→ c, 2→ a, 3→ b, 4→ a
	Lesson 2 :	1→ a, 2→ c
	Lesson 4 :	1→ a, 2→ b
	Lesson 5 :	1→ a, 2→ c
	Lesson 6 :	1→ c, 2→ d, 3→ a, 4→a
	Lesson 7 :	1→ a, 2→ c
	Lesson 8 :	1→ b, 2→ a, 3→ c, 4→ a
	Lesson 9 :	1→ a, 2→ a, 3→ b, 4→ c, 5→ a
Unit 8 :	Lesson 1 :	1→ a, 2→ b, 3→ c, 4→d
	Lesson 2 :	1→ b, 2→ d, 3→ b, 4→ a
Unit 9 :	Lesson 1 :	1→ a, 2→ a, 3→ b, 4→ a, 5→ a
	Lesson 2 :	1→ a, 2→ b, 3→d