**Pointers**

A pointer is a variable. It contains the memory address of another variable which is declared as a pointer type.

int a = 5

5

2000

\*\* The ‘&’ operator is called the address operator. It represents the address of a variable.

#include <stdio.h>

main()

{

int a = 5;

printf (“value of a = %d”,a);

printf (“address of a = %u”,&a);

}

The %u is used as a format specifier for obtaining address.

**Output:**

Value of a = 5

Address of a = 2000

a

Variable name

5

Value at address 2000

2000

address

**\* operator:** value at address operator. It represents the value at the specified address.

#include <stdio.h>

main()

{

int a = 5;

printf (“value of a = %d”,a);

printf (“address of a = %u”,&a);

printf (“value at address = %u = %d”,&a, \*(&a));

}

**Output:**

Value of a = 5

Address of a = 2000

Value at address = 2000 = 5

\*(&a) = a (means the value at address of variable a)

**Declaration of Pointers:**

When a pointer variable is declared then an asterisk (\*) symbol should precede the variable name.

int a = 5;

int \*b;

Example 3:

#include <stdio.h>

main()

{

int a = 5;

int \*b;

b = &a;

printf (“value of a = %d”,a);

printf (“value of a = %d”,\*(&a));

printf (“value of a = %d”,\*b);

printf (“address of a = %u”,&a);

printf (“address of a = %u”,b);

printf (“address of b = %u”,&b);

printf (“value of b = address of a = %u”, b);

}

Example 4:

#include<stdio.h>

main ()

{

char a = ‘p’;

char \*a1;

int b = 5;

int \*b1;

float c = 5.5;

float \*c1;

a1 = &a;

b1 = &b;

c1 = &c;

printf(“value of a = %c\n”,a);

printf(“value of a = %c\n”,\*(&a));

printf(“value of a = %c\n”,\*a1);

printf(“value of b = %d\n”,b);

printf(“value of b = %d\n”,\*(&b));

printf(“value of b = %d\n”,\*b1);

printf(“value of c = %f\n”,c);

printf(“value of c = %f\n”,\*(&c));

printf(“value of c = %f\n”,\*c1);

printf(“address of a = value of a1= %u\n”,a1);

printf(“address of b = value of b1= %u\n”,b1);

printf(“address of c = value of c1= %u\n”,c1);

}

**Pointer variable stores the address of the first byte.**

**POINTER to POINTER**

#include<stdio.h>

main()

{

int a = 5;

int \*b;

int \*\*c;

b = &a;

c = &b;

printf(“value of a = %d\n”,a);

printf(“value of a = %d\n”,\*(&a));

printf(“value of a = %d\n”,\*b);

printf(“value of a = %d\n”,\*\*c);

printf(“value of b = address of a = %u\n”,b);

printf(“value of c = address of b = %u\n”,c);

printf(“address of a = %u\n”,&a);

printf(“address of a = %u\n”,b);

printf(“address of a = %u\n”,\*c);

printf(“address of b = %u\n”,&b);

printf(“address of b = %u\n”,c);

printf(“address of c = %u\n”,&c);

}

Note: Here ‘b’ is a pointer variable which contains the address of variable ‘a’. ‘c’ is a pointer to a pointer variable, which contains the address of the pointer variable ‘b’

**Pointer Arithmetic:**

In pointers postfix, prefix, increment, decrement means addition or subtraction of bytes that pointer data type holds, with the value that pointer variable contains.

Ex-

int a = 5;

int \*b;

b = &a;

a b

5

1000

1000 2000

Here b++ or ++b means 1000+2 = 1002

And b-- or --b means 1000-2 = 998

int a = 5;

a++ or ++a = 5+1 =6

a-- or --a = 5-1 = 4

#include<stdio.h>

main()

{

int a = 5;

int \*b;

b = &a;

printf(“address of a = %u\n”,b); 1000

printf(“value of b = %u\n”,++b); 1002

printf(“value of b = %u\n”, b++); 1002

printf(“value of b = %u\n”,--b); 1002

printf(“value of b = %u\n”, b--); 1002

printf(“value of b = %u\n”, b); 1000

}

++b ------🡪 1st increment then print

b++ ----🡪 1st print then increment

Ex –

#include<stdio.h>

main()

{

int a = 5;

int \*a1;

char b = ‘x’;

char \*b1;

float c = 5.5;

float \*c1;

a1 = &a;

b1 = &b;

c1 = &c;

printf(“ address of a = value of a1 = %u\n”,a1);

printf(“ address of b = value of b1 = %u\n”,b1);

printf(“ address of c = value of c1 = %u\n”,c1);

a1++;

b1++;

c1++;

printf(“ Now value of a1 = %u\n”,a1); 1002

printf(“ Now value of b1 = %u\n”,b1); 4001

printf(“ Now value of c1 = %u\n”,c1); 8004

}

a1 = 1000

b1 = 4000

c1 = 8000

1. Addition of a number to a pointer variable

Ex-

int a = 5;

int \*b;

b = &a;

b = b+1;

b = b+2;

a = 5

b = 1000

b = b+1 means 1000+(1\*2 )= 1000+2 = 1002

b = b+2 means 1000+(2\*2) = 1000+4 = 1004

1. Subtraction of a number from a pointer variable

Ex –

int a = 5;

int \*b;

b = &a;

b = b-1;

b = b-3;

b = b-1 = 1000-(1\*2) = 1000-2 =998

b = b-3 = 1000-(3\*2) = 1000-6 =994

**(111) Subtraction of one pointer from another pointer ---🡪 array**

\*\*These operations can never be done with pointers:

1. Addition between two pointers.
2. Multiplication between pointer and any number.
3. Division between pointer and any number.

**POINTER and FUNCTIONS –**

1. Call by value
2. Call by reference

/\*program to explain call by value\*/

#include<stdio.h>

main()

{

int a =5;

int b =8;

printf(“ Before calling the function a and b are %d,%d\n”,a,b); **a =5, b = 8**

value (a,b);

printf(“ After calling the function a and b are %d,%d\n”,a,b”); **a =5, b = 8**

}

value ( p, q)

int p, q;

{

p++;

q++;

printf(“ In function changes are %d,%d\n”,p,q); **p = 6, q = 9**

}

/\* program to explain call by reference \*/

#include<stdio.h>

main()

{

int a = 5;

int b = 8;

printf(“ Before calling the function a and b are %d,%d\n”,a,b); **a= 5, b = 8**

ref (&a, &b);

printf(“ After calling the function a and b are %d,%d\n”,a,b”); **a = 6 and b = 9**

}

ref (p, q)

int \*p,\*q;

{

(\*p)++;

(\*q)++;

printf(“ In function changes are %d,%d\n”,\*p,\*q); **\*p = 6, and \*q = 9**

}

We can declare the pointer function which can return an address. But we should declare the function as a pointer data type.

**#include <stdio.h>**

**main()**

**{**

**int a = 5;**

**int \*b;**

**int \*fun();**

**b = &a;**

**printf (“Before calling the function address of a = %u\n”,b); 1000**

**b = fun(&a);**

**printf (“After calling the function value of b = %u\n”,b); 1002**

**}**

**int \*fun(int \*\*x)**

**{**

**\*x = \*x+1;**

**return (\*x);**

**}**

**a = 5**

**b = 1000**