

TATA ELXSI

engineering creativity

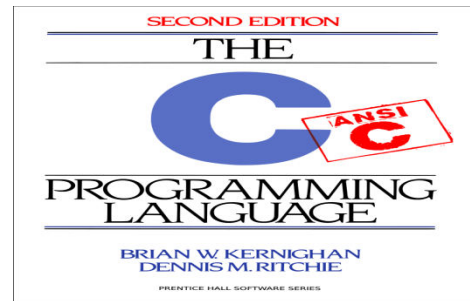
C Programming

Day - 3



disclaimer

- <http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html>
- ANSI C K&R



Day – 3 Agenda

- What Is a Pointer
- Declaring Pointers
- Initializing Pointers
- Pointers and Variable Types
- Pointers and Arrays
- Pointer Arithmetic
- Incrementing Pointers
- Other Pointer Manipulations
- Passing Arrays to Functions
- Pointer to pointer
- Pointer arithmetic with pointer to pointer

Pointers



Objectives

- What Is a Pointer
- Declaring Pointers
- Initializing Pointers
- Pointers and Variable Types
- Void pointers
- Pointers and Arrays
- Pointer Arithmetic
- Incrementing Pointers
- Other Pointer Manipulations
- Passing Arrays to Functions

What Is a Pointer ?

A PC's RAM consists of many thousands of sequential storage locations, and each location is identified by a unique address.

When declaring a variable in a C program, the compiler sets aside a memory location with a unique address to store that variable.

When the program uses the variable name, it automatically accesses the proper memory location. The location's address is used, but it is hidden.

To summarize, a pointer is a variable that contains the address of another variable. Now here are some details of using pointers in the C programs.

Declaring Pointers

A pointer is a numeric variable and, like all variables, must be declared before it can be used.

A pointer declaration takes the following form:

typename *ptrname;

Example

```
char *ch1, *ch2;
```

```
float *value=NULL, percent = 3.1415f;
```


Initializing Pointers

Until a pointer holds the address of a variable, it isn't useful.

the program must put it there by using the address-of operator, the ampersand (&).

When placed before the name of a variable, the address-of operator returns the address of the variable.

Therefore, initialize a pointer with a statement of the form
pointer = &variable;

Pointers and Variable Types

- The address of a variable is actually the address of the first (lowest) byte it occupies.
- `int vint = 12252;`
- `char vchar = 90;`
- `float vfloat = 1200.156004;`
- `int *p_vint;`
- `char *p_vchar;`
- `float *p_vfloat;`
- `p_vint = &vint;`
- `p_vchar = &vchar;`
- `p_vfloat = &vfloat;`

```
# include <stdio.h>
```

```
main()
```

```
{
```

```
    int  x = 5;
```

```
    int  *p = &x;
```

```
    printf("x=%d\n",x);
```

```
    printf("p=%p\n",p);
```

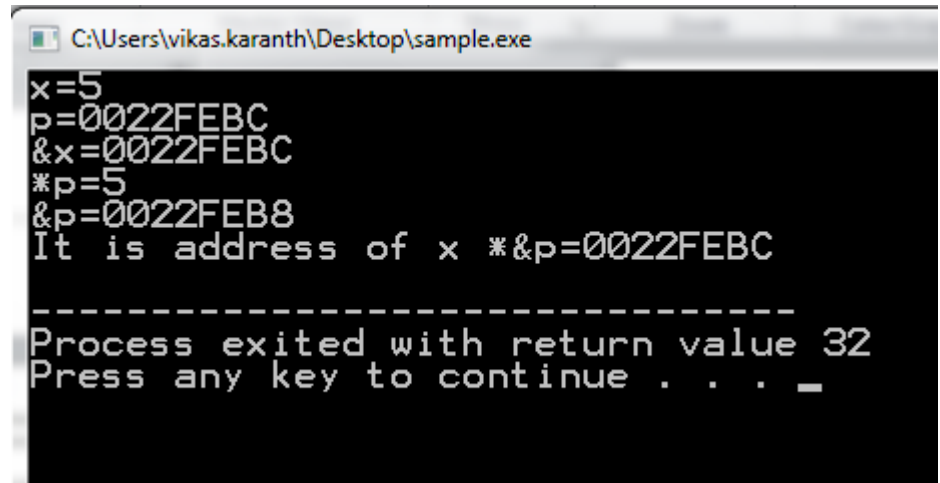
```
    printf("&x=%p\n",&x);
```

```
    printf("*p=%d\n",*p);
```

```
    printf("&p=%p\n",&p);
```

```
    printf("It is address of x *&p=%p\n",&p);
```

```
}
```



```
C:\Users\vikas.karanth\Desktop\sample.exe
x=5
p=0022FEBC
&x=0022FEBC
*p=5
&p=0022FEB8
It is address of x *&p=0022FEBC

-----
Process exited with return value 32
Press any key to continue . . . _
```

void pointers

Pointers defined to be of a specific data type cannot hold the address of some other type of variable.

This problem can overcome by using a general purpose pointer type called void pointer.

E.g. `void *v_ptr;`

Pointers defined in this manner do not have any type associated with them and can hold the address of any type of variable.

Pointers to void cannot be directly dereferenced like other pointer variables using the indirection operator. Prior to dereferencing a pointer to void, it must be suitably typecast to the required data type.

Syntax : *((data type *))pointervariable;

E.g. void pointer

main()

```
{  
    int i1=100;  
    float f1=200.5;  
    void *vptr ;  
    vptr = &i1;  
    printf(" i1 contains %d",*((int *)vptr));  
}
```

Pointers and Arrays

An array name is a pointer to the array's first element.

Using the expression `&data[0]` to obtain the address of the array's first element is allowed.

In C, the relationship `(data == &data[0])` is true.

The name of an array is a pointer to the array (this is a pointer constant).

it can't be changed and remains fixed for the duration of program execution.

Pointers and Arrays

- `int *pa;` then the assignment
- `pa = &a[0];`
- sets `pa` to point to element zero of `a`; that is, `pa` contains the address of `a[0]`.

`a[i]` -> is internally evaluated as :Base address + index * sizeof(data_type);

- Thus `a[5]` :
- By considering base address(2000) which is evaluated as
- $2000 + 5 * 4 = 2020$ -> `cof(2020)`

Pointers and Arrays

```
#include <stdio.h>
int my_array[] = {1,23,17,4,-5,100};

int *ptr;
int main(void)
{
    int i; ptr = &my_array[0]; /* point our pointer to the first element of the array */
    printf("\n\n");
    for (i = 0; i < 6; i++) {
        printf("my_array[%d] = %d ",i,my_array[i]); /*<-- A */
        printf("ptr + %d = %d\n",i, *(ptr + i)); /*<-- B */
    }
    return 0;
}
```


Pointers and Arrays

```
#include <stdio.h>
/* bubble sort using pointers */
main()
{
    int num[10],i,j,t;
    int *p=num;
    printf("enter elements into the array\n");
    for(i=0;i<5;i++)
        scanf("%d",p+i);
    for(i=0;i<5;i++)
        for(j=0;j<5-1;j++)
        {
```

Pointers and Arrays

```
if (*(p+j) > *(p+(j+1)))
{
    t=*(p+j);
    *(p+j)=*(p+(j+1));
    *(p+(j+1))=t;
}
}
puts("printing .....\\n");
for(i=0;i<5;i++)
    printf("%d\\n", *(p+i));
}
```

Thank you



TATA ELXSI

ITPB Road Whitefield
Bangalore 560 048 India
Tel +91 80 2297 9123
Fax +91 80 2841 1474
e-mail info@tataelxsi.com

www.tataelxsi.com