Pointer

Structured Programming Language (CSE-1271)

Course Instructor : Mohammed Mamun Hossain
Assistant Professor, Dept. of CSE, BAUST

Outline

1. Introduction
2. Variables and Memory
3. Indirection
4. Base Type
5. Pointer as Array

Pointer

- A pointer is a variable that holds the memory address of another object (variable).

- If a variable called **p** contains the address of another variable called **q**, then p is said to point to q.

- Then if **q** is at location 100 (say) in memory, then **p** would have the value 100.

Variable and Memory Address

- p

Variable	RAM	Value
q	1024	A
a	1025	108
	1026	
	1027	
	1028	
	1029	unknown

```
char q;
int a;

q = 'A';

a = 108;
```

How to Declare?

General form:

```
type *var-name;
```

Here, type is the base type of the pointer

It specifies the type of the object that the pointer can point to.

The asterisk (*) tells the computer that a pointer variable is being created.

Example:

```
int *p;
```

Variable and Memory Address

p

Variable	RAM	Value
q	1024	A
p	1025	1024
	1026	unknown
	1027	unknown
	1028	unknown
	1029	unknown

```
char q;
Char *p;
q = 'A';
P = &q;
```

Poi

We can verbalize & as address of

We can verbalize * as at address

```
4   int main()
5   {
6       int *p, q;
7
8       q = 199; // Assign
9
10      p = &q;  // Assign
11
12      printf("\n Value of *p : %d", *p);
13      // display q's value using
14      printf("\n Value of  p : %d\n\n", p);
15      // value
16
17      return 0;
18  }
```

"E:\C Code\Teach Yourself C\Page - 167\bin\Debug\Page - 167.exe"

Value

Defines two variable: **p** is integer pointer and **q** is integer

Assign **q** the value 199

Assign **p** the address of **q**

Print the value at address **q**

Print the value of **p** (it holding the address of **q**)

Pointer | Example 2

```c
#include <stdio.h>

int main()
{
    int *p, q;

    p = &q;  // Assign p the address of q

    *p = 199; // Assign q a value using a pointer

    printf("\n q's Value of is %d\n\n\n", q);

    return 0;
}
```

"E:\C Code\Teach Yourself C\Page - 168\bin\Debug\Page - 168.exe"

```
q's Value of is 199
```

Pointer | Indirection

- When a variable's value is referenced through a pointer, the process is called indirection.

```
int *p, q;
```

Location	Contents
100 (p)	Unknown
104 (q)	Unknown

```
p = &q;
```

Location	Contents
100 (p)	104
104 (q)	Unknown

p points to q

```
*p = 199;
```

Location	Contents
100 (p)	104
104 (q)	199

p points to q

Pointer | Base type

- The base type of a pointer is very important.
- Though C allow any type of pointer to point anywhere in memory, but the base type determine how the object pointed to will be treated.
- The C compiler uses the base type to determine how many bytes are in the object pointed to by the pointer.
- This is how the compiler now how many bytes to copy when an indirection assignment is made

Pointer | Base type

- Consider this fragment:
- not syntactically incorrect
- but wrong
- fp assign address of an integer
- this address try hold floating point value
- int is shorter than double

```
int q;
double *fp;

fp = &q;

*fp = 200.54;
```

Pointer | Another Example

- If we attempt to use a pointer before it has been assign the address of a variable - our program probably crash.

```
int *p;

*p = 10;
```

- A pointer that contains a null value (zero) is assumed to be unused and pointing at nothing.

- In C, a null is assumed to be an invalid memory address.

Pointer | Restrictions

- In addition to “*” and “&” operators, it supports only four other operators-

+	++	-	--

- We only can add or subtract

only integer quantities.

Pointer | Use Array by Pointer

- In C, pointers and arrays are closely related.
- They are often interchangeable
- This relationship between the two makes their implementation both unique and powerful

Pointer | Use Array by Pointer

❖ When we use an array name without an index, we are generating a pointer to the start of the array.

```
ain.c ×
2   #include <stdlib.h>
3
4   int main()
5   {
6       char str[80];
7       int i;
8
9       printf("Enter a
10      gets(str);
11
12      //for(i=0;str[i],i++)
13      printf(str);
14      printf("\nUsing format specifier: %s\n\n\n",str);
15
16      return 0;
17  }
```

- Here what is being passed to gets() is not an array but pointer
- **Actually we can't pass an array to a function in C we only pass a pointer**
- The gets() function uses the pointer to load the array it points to with the characters we enter at the keyboard

Pointer | Use Array by Pointer

- Since the array name **without an index** is a pointer to the start of the array.

- it is possible to **assign that value to another pointer** and access the array **using pointer arithmetic**.

Pointer | Use Array by Pointer

```
"E:\C Code\Teach Yourself C\Chapter 6 - poiter\pag
10 20 30
10 20 30
```

```c
int main()
{
    int a[10] = {10,20,30,40,50,60,70,80,90,100};
    int *p;

    p = a;

    printf("%d %d %d\n",*p,*(p+1),*(p+2));

    printf("%d %d %d\n",a[0],a[1],a[2]);

    return 0;
}
```

Assign p the address of start of a

Similar as p = &q

This print the first, second and third elements of array a

Print these element using array index

Pointer | Use Array by Pointer

```
printf("%d %d %d\n",*p,*(p+1),*(p+2));
```

- Here the parentheses in expressions such as *(p+1) are necessary because the * has higher precedence than the + operator

Pointer | Use Array by Pointer

- We can index a pointer as if it were an array

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

int main()
{
    char str[]="Pointers are fun!!!";
    char *p;
    int i;

    p = str;

    for(i=0; str[i]; i++)
    printf("%c",p[i]);

    return 0;
}
```

```
"E:\C Code\Teach Yourself C\Chapter 6 - poiter\page 178\main.exe"
Pointers are fun!!!
Process returned 0 (
```

Keep one point firmly in your mind: We should index a pointer only when that pointer points to an array.

```c
int main()
{
    char *p, ch;
    int i;

    p = &ch;

    for(i=0; i<10; i++)
        p[i]='A'+i; //Wrong!

    return 0;
}
```

Wrong: Since ***ch*** is not an array. It can't meaningfully indexed.

Pointer | Use Array by Pointer

- We can use pointer arithmetic rather than array indexing to access elements to the array.

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

int main()
{
    char str[80];

    *(str+3) = 'c';
    printf("Using pointer    : %c", *(str+3));
    printf("\nUsing array index: %c\n", str[3]);

    return 0;
}
```

"E:\C Code\Teach Yourself C\Chapter 6 - poiter\page 179\main.exe"

```
Using pointer    : c
Using array index: c
```

- **Remember! We can't modify the value of the pointer generated by using an array name.**
- **In this example str++ is wrong.**

Pointer | Multiple indirection

- When a pointer point to another pointer, it is called **multiple indirection**.
- When a pointer points to another pointer, the first pointer contain the address of the second pointer, which is location containing the object.
- To declare a pointer to a pointer, an additional asterisk is placed in front of the pointers name – char **mp;

Pointer | Multiple indirection

```c
#include<stdio.h>

int main()
{
    char **mp, *p, ch;
    p=&ch;
    mp=&p;
    **mp='Y';
    printf("Address of\n ch= %d\n  p= %d\n mp= %d\n\n",&ch,&p,&mp);
    printf("Values of\n  ch= %c\n  p = %d\n  mp= %d\n",ch,p,mp);
    printf("  *p= %c\n *mp= %d\n**mp= %c\n",*p,*mp,**mp);

    return 0;

}
```

"D:\MyWork\programming contest\

```
Address of
 ch= 2752263
  p= 2752264
 mp= 2752268

Values of
  ch= Y
  p = 2752263
  mp= 2752264
  *p= Y
 *mp= 2752263
**mp= Y
```

Please try to understand the code and output

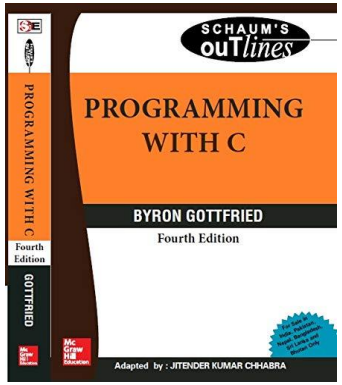
Thank You.

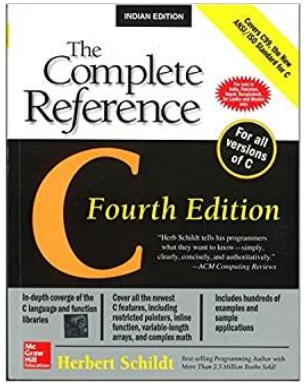
Questions and Answer

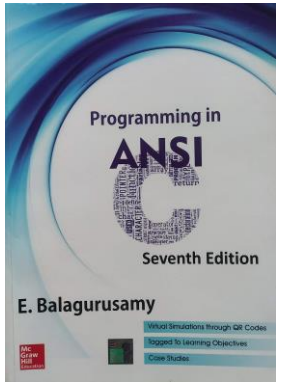
References

Books:

1. Programming With C. *By Byron Gottfried*
2. The Complete Reference C. *By Herbert Shield*
3. Programming in ANSI C *By E. Balagurusamy*
4. Teach yourself C. *By Herbert Shield*







Web:

1. www.wikbooks.org

and other slide, books and web search.