Computer Science & IT

C Programming

String & Structure

Lecture No. 02

By- Abhishek Sir

Recap of Previous Lecture

- Topic — Strong declaration
- Topic — using character array — Mutable
- Topic — using character pointer — Immutable (ROM) Code Area
- Topic
- Topic

Slide

Topics to be Covered

- Topic — Array of string
- Topic — Structure
- Topic — Union
- Topic
- Topic

Slide

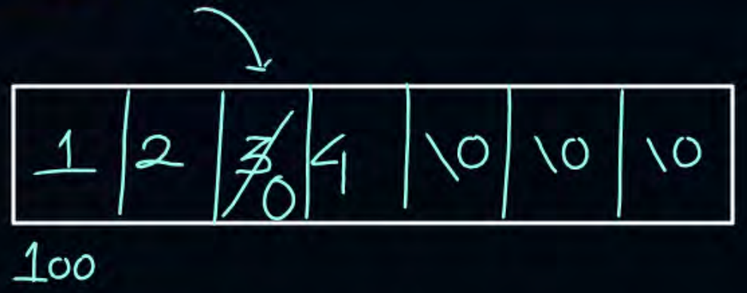
Question

Consider the following C program segment.

```
# include <stdio.h>
int main(){
    char s1[7] = "1234", *p;
    p = s1 + 2;
    *p = '0';
    printf("%s", s1);
}
```

What will be printed by the program?

(A) 12
(B) 120400
(C) 1204
(D) 1034



p = s1+2 = 100+2 = 102

'0' Zero character 48

'\0' NULL character 0

Slide

Question

Consider the following function written in the C programming language.

```
void foo(char *a){
        if ( *a && *a != '\n'){
            foo(a+1);
            putchar(*a);
        }
    }
```

The output of the above function on input "ABCD EFGH" is

(A) ABCD EFGH
(B) ABCD
(C) HGFE DCBA
(D) DCBA

getchar takes a character from Input

putchar output a character

Slide

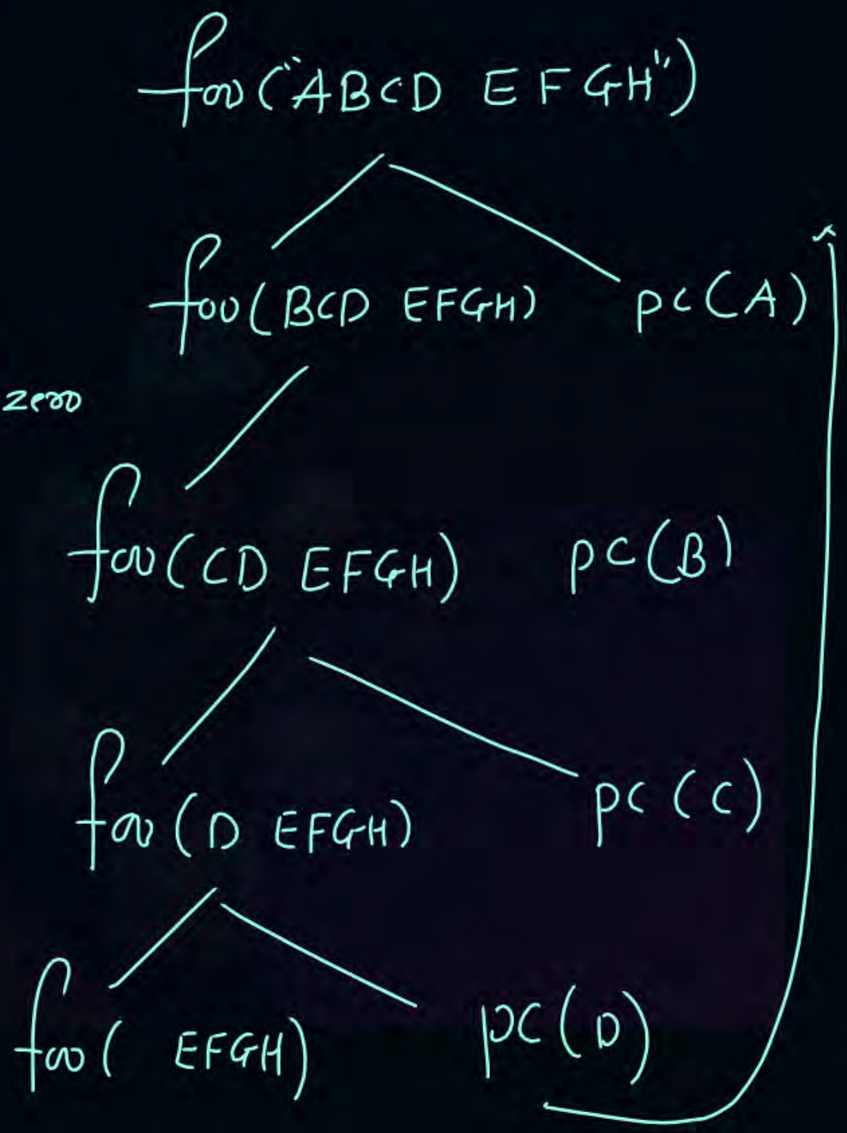
Consider the following function written in the C programming language.

```
void foo(char *a){
        if ( *a && *a != ' '){
            foo(a+1);
            putchar(*a);
        }
}
```

*a && *a != ' ' → Space 32

*a && *a != ' '

NOT NULL + NULL / Not zero



The output of the above function on input "ABCD EFGH" is

(A) ABCD EFGH
(B) ABCD
(C) HGFE DCBA
(D) DCBA

lide

Array of character pointer

```
printf("%c", *ch[0]);
printf("%c", *(ch[1]+1));
```

`*ch[0]` → `ch[0][0]`

`*(ch[1]+1)` → `ch[1][1]`

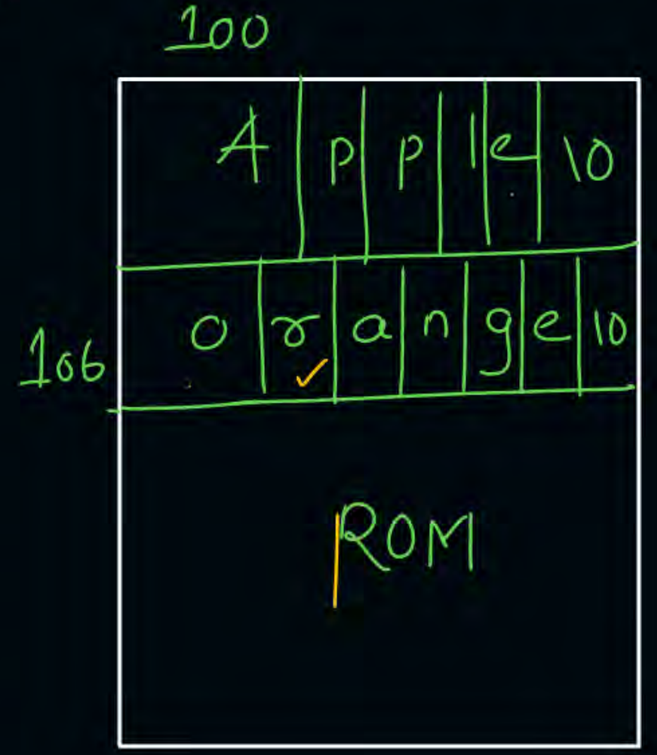
```
Char *ch[] = { "Apple", "orange" };
```

Array of string

*ch[]	100	106
	200	208

String print:

```
printf("%s", ch[0]);
printf("%s", *ch);
printf("%s", ch[1]);
printf("%s", *(ch+1));
```



Strlen

```
#include <stdio.h>
int main() {
    char *ch[2] = {"parakram", "vijay"};
     printf("%s\n",*ch+1);
     return 0;
}
```

(A) parakram
(B) arakram
(C) vijay
(D) ijay

100	109
200	208

*ch + 1
*(200)
100 + 1 : 101

Name of array Address of first element

p	a	r	a	k	r	a	m	\0	v	i	j	a	y	\0
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114

Slide

Strlen

```
#include <stdio.h>
int main() {
    char *ch[2] = {"parakram", "vijay"};
     printf("%s\n",*(ch+1)+1);
     return 0;
}
```

(A) parakram
(B) arakram
(C) vijay
(D) ijay

*(ch+1) = (200+1)

= *(208)

ch[1] = 109

= 109+1 = 110

p	a	r	a	k	r	a	m	\0	v	i	j	a	y	\0
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114

Strlen

```
#include <stdio.h>
int main() {
    char *ch[2] = {"parakram", "vijay"};
    printf("%d\n",ch[1][3]);
    return 0;
}
```

Output of the program is _____97_____

ch[1][3]

((ch+1)+3)

((200+1)+3)

((208)+3)

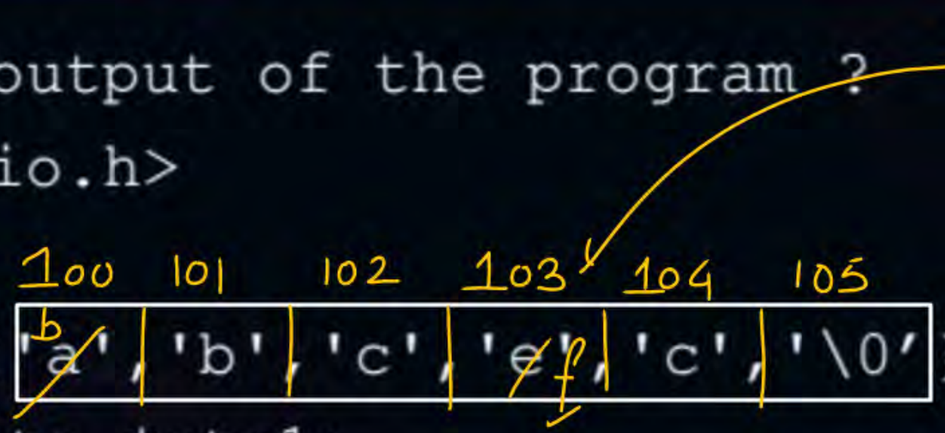
*(109+3) = (112)

p	a	r	a	k	r	a	m	\0	v	i	j	a	y	\0
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114

Question

What is the output of the program ?

```
#include<stdio.h>
int main(){
   char s[]={'a','b','c','e','f','c','\0'};
   char *p,*str,*str1;
   p=&s[3];
   str=p;
   str1=s;
   printf("%d",++*p + ++*str1-32);
   return 0;
}
```



P 103

str 103

str1 100

a 97
b 98
c 99
d 100
e 101
f 102

102 + 98 - 32

200 - 32

168

Slide

user defined data type structure

Structur is collection of dissimilar data type

```
struct test {
        int a,
        char ch.
};
```

No memory allocation

User defined data type Structure

Structur is collection of dissimilar data type

```
struct test {
        int a,
        char ch.
} u;
```

u is variable

```
struct test u1;
struct test u1 = { 10, 'a' },
u1.a = 11;
u1.ch = 'b';
```

user defined data type structure

Structur is collection of dissimilar data type

```
typedef struct test {
        int a,
        char ch.
} u;
```

u is alias

```
struct test t1;
u t1
```

t1 is a structure variable

user defined data type structure

Structur is collection of dissimilar data type

given Name is Not mandatary

```
typedef struct {
    int a,
    char ch.
} u;
```

User defined data type Structure

Structur is collection of dissimilar data type

```
struct test {
    int a,
    char ch,
},
```

```
struct test t = { 20, 'x' };
struct test *ptr;
ptr = &t;
printf("%d %d", ptr->a,
                t.a);
```

To Access Member using Structure

pointer → operator is used

Question

Q The following C declarations

```
struct node{
        int i;
        float j;
};
struct node *s[10];
```

define s to be

*s[10]

s - array of structure of type node

each element is an Addr

(a) An array, each element of which is a pointer to a structure of type node

(b) A structure of 2 fields, each field being a pointer to an array of 10 elements

(c) A structure of 3 fields: an integer, a float, and an array of 10 elements

(d) An array, each element of which is a structure of type node

create pointer to an int array with 10 elements:

int (*ptr)[10]

ptr is pointer to array of 10 integer

int a[10];
ptr = &a;

Slide

Question

Consider the following C declaration

```
struct {
        short s [5]
        union {
        float y;
        long z;
        } u;
}t;
```

Assume that objects of the type short, float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The memory requirement for variable t,

(a) 22 bytes
(b) 14 bytes
(c) 18 bytes
(d) 10 bytes

short : 2B - x5 : 10B
float - 4B
long = 8B
max
8B
10+8 : 18B

structure
struct test{
int a,
char ch;
};
5B

union test1 t2,
t2
max(int, char)
union 4B
union test1 {
int a;
char ch.
}

a | ch
4B | 1B
Struct test t1,

Slide

Question

Consider the following C declaration

```
struct {
        short s [5]
        union {
        float y;
        long z;
        } u;
}t;
```

float y to be assigned with 3.14

t.u.y = 22.0/7.

Assume that objects of the type short, float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The memory requirement for variable t,

(a) 22 bytes
(b) 14 bytes
(c) 18 bytes
(d) 10 bytes

Slide

Question

Consider the following C program

```
#include<stdio.h>
struct Ournode{
char x,y,z;
};
int main(){
struct Ournode p =  {'1' ,'0', 'a' +2};
struct Ournode *q = &p;
printf("%c,  %c",    *((char*)q+1),    *((char*)q+2));
return 0;
}
```

The output of this program is:

(A) 0, c ✓

(B) 0, a+2 ✗

(C) '0', 'a+2' ✗

(D) '0', 'c'

0

97+2 = 99

1	0	C
100	101	102

*(100+1)

*(101)

*(100+2)

= *(102) = C

Slide

2 mins Summary

Topic

array of string

Topic

structure

Topic

practice problem

Topic

Topic

Slide

THANK - YOU