CSE102
Computer Programming with C

2016-2017 Fall Semester

Structures

Examples and Teasers

© 2015-2016 Shahıd Alam

Examples & Teasers 1 & 2

```
typedef struct
{
        int a;
        char b;
} S;

typedef union
{
        int a;
        char b;
} U;
```

Examples & Teasers 1

```
S s;
U u;

s.a = 10;
u.a = 10

s.b = 'A';
u.b = 'A';

printf("s.a: %d u.a: %d\n", s.a, u.a);
printf("s.b: %c u.b: %c\n", s.b, u.b);
```

Examples & Teasers 1

```
S s;
U u;

s.a = 10;
u.a = 10

s.b = 'A';
u.b = 'A';

printf("s.a: %d u.a: %d\n", s.a, u.a);
printf("s.b: %c u.b: %c\n", s.b, u.b);

OUTPUT:
s.a: 10 u.a: 65
s.b: A u.b: A
```

Examples & Teasers 2

```
S s;
U u;

s.a = 0x0ABCDEFA;
u.a = 0x0ABCDEFA;

s.b = 0x41;
u.b = 0x41;

printf("s.a: %X u.a: %X\n", s.a, u.a);
printf("s.b: %c u.b: %c\n", s.b, u.b);
```

Examples & Teasers 2

```
S s;
U u;

s.a = 0x0ABCDEFA;
u.a = 0x0ABCDEFA;

s.b = 0x41;
u.b = 0x41;

printf("s.a: %X u.a: %X\n", s.a, u.a);
printf("s.b: %c u.b: %c\n", s.b, u.b);

OUTPUT:
s.a: ABCDEFA u.a: ABCDE41
s.b: A u.b: A
```

Examples & Teasers 3

Byte ordering - How to check the endianness of a machine (use Union)

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	
1001	
1002	
1003	

Address	Value
1000	
1001	
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	
1002	
1003	

Address	Value
1000	
1001	
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	
1003	

Address	Value
1000	
1001	
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	0C
1003	

Address	Value
1000	
1001	
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

`The number stored:`
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	0C
1003	0D

Address	Value
1000	
1001	
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored: $0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	0C
1003	0D

Address	Value
1000	0D
1001	
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	0C
1003	0D

Address	Value
1000	0D
1001	0C
1002	
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	0C
1003	0D

Address	Value
1000	0D
1001	0C
1002	0B
1003	

Examples & Teasers 3

Big Endian: In big endian, the *most significant byte* is stored in the smallest address.

The number stored:
$0A0B0C0D_{16}$

Little Endian: In little endian, the *least significant byte* is stored in the smallest address.

Address	Value
1000	0A
1001	0B
1002	0C
1003	0D

Address	Value
1000	0D
1001	0C
1002	0B
1003	0A

Examples & Teasers 3

Byte ordering - How to check the endianness of a machine (use Union)

```
union
{
    int i;
    char c[sizeof(int)];
} un;

un.i = 0x0A0B0C0D;

if (un.c[0] == 0xA && un.c[1] == 0xB)
    printf("big-endian\n");
else if (un.c[0] == 0xD && un.c[1] == 0xC)
    printf("little-endian\n");
else
    printf("unknown\n");
```

Examples & Teasers 4

```
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int d;
int array[] = {23,34,12,17,204,99,16};
```

```
for (d = -1; d <= TOTAL_ELEMENTS-2; d++)
      printf("%d\n",array[d+1]);
```

OUTPUT: ??? ???

Examples & Teasers 4

```
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int d;
int array[] = {23,34,12,17,204,99,16};

d = -1;
printf("--%d--\n", d <= TOTAL_ELEMENTS-2);
printf("--%d--\n", d <= 7-2);

for (d = -1; d <= TOTAL_ELEMENTS-2; d++)
      printf("%d\n",array[d+1]);
```

OUTPUT: ??? ???

Examples & Teasers 4

```
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int d;
int array[] = {23,34,12,17,204,99,16};

d = -1;
printf("--%d--\n", d <= TOTAL_ELEMENTS-2);
printf("--%d--\n", d <= 7-2);

for (d = -1; d <= TOTAL_ELEMENTS-2; d++)
      printf("%d\n",array[d+1]);
```

NOTE: sizeof returns unsigned int

OUTPUT: ??? ???

Examples & Teasers 4

```
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int d;
int array[] = {23,34,12,17,204,99,16};

d = -1;
printf("--%d--\n", d <= TOTAL_ELEMENTS-2); // prints 0
printf("--%d--\n", d <= 7-2); // prints 1

for (d = -1; d <= TOTAL_ELEMENTS-2; d++)
      printf("%d\n",array[d+1]);
```

NOTE: sizeof returns unsigned int

OUTPUT:

```
--0--
--1--
```

Examples & Teasers 4

```
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int d;
int array[] = {23,34,12,17,204,99,16};

d = -1;
printf("--%d--\n", d <= TOTAL_ELEMENTS-2); // prints 0
printf("--%d--\n", d <= 7-2); // prints 1

for (d = -1; d <= TOTAL_ELEMENTS-2; d++)
      printf("%d\n",array[d+1]);
```

NOTE: sizeof returns unsigned int

```
OUTPUT:   ??? FIX IT ???
--0--
--1--
```

Examples & Teasers 4

```
#define TOTAL_ELEMENTS (sizeof(array) / sizeof(array[0]))

int d;
int array[] = {23,34,12,17,204,99,16};

d = -1;
printf("--%d--\n", d <= TOTAL_ELEMENTS-2); // prints 0
printf("--%d--\n", d <= 7-2); // prints 1

for (d = 0; d < TOTAL_ELEMENTS; d++)
      printf("%d\n",array[d]);
```

NOTE: sizeof returns unsigned int

```
OUTPUT:   ??? FIXED  ???
--0--
--1--
```

Examples & Teasers 5

```
int i, k;
for (i = 0; i < 12; i++)
{
        if (i % 2 == 0)
        {
                for (k = 0; k < i; k = k + 2)
                        printf("%d ", k);
        }
}
printf("\n");
```

Examples & Teasers 5

What is printed as a result of executing the code segment in the previous slide?

(A) 0 2 0 2 0 2 0 2

(B) 0 0 2 0 2 4 0 2 4 6 0 2 4 6 8

(C) 0 0 2 0 0 2 0 0 2 0 0 2

(D) 0 2 0 2 0 2 0 2 0 2

(E) 0 1 2 0 1 2 3 4 0 1 2 3 4 5

Examples & Teasers 5

What is printed as a result of executing the code segment in the previous slide?

(A) 0 2 0 2 0 2 0 2

(B) 0 0 2 0 2 4 0 2 4 6 0 2 4 6 8

(C) 0 0 2 0 0 2 0 0 2 0 0 2

(D) 0 2 0 2 0 2 0 2 0 2

(E) 0 1 2 0 1 2 3 4 0 1 2 3 4 5