

CM 7: Dynamic Memory allocation

Arrays

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int r = 3, c = 4;
    int *arr = (int *)malloc(r * c * sizeof(int));
    int i, j, count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            ??? = ++count;
    return 0;
}
```

■ How we initialize the array?

- `*(arr + i*c + j)`
- `arr[i][j]`
- `arr + i*c + j`
- `*(&arr[i]+&arr[j])`

Arrays

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

int main(){
    int r = 3, c = 4, i, j, count;
    int *arr[r];
    for (i=0; i<r; i++)
        arr[i] = (int *)malloc(c * sizeof(int));
    count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            ????? = ++count;
    return 0;
}
```

■ How we initialize the array?

- `*(arr+i*c+j)`
- `*( *(arr + i*c) + j)`
- `*(*(arr + i*c + j))`
- `*(*(arr+i)+j)`

3

Arrays

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

int main(){
    int r = 3, c = 4, i, j, count;

    int **arr = (int **)malloc(r * sizeof(int *));
    for (i=0; i<r; i++)
        arr[i] = (int *)malloc(c * sizeof(int));

    count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            arr[i][j] = ++count;

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", arr[i][j]);

    free(arr);

    return 0;
}
```

■ Is the code correct?

- Yes
- No

L3Info – Unix/C

4

Problems

■ Match the following

X: `m=malloc(5); m= NULL;` 1: using dangling pointers
Y: `free(n); n->value=5;` 2: using uninitialized pointers
Z: `char *p; *p = 'a';` 3: lost memory

```
#include<stdio.h>
int main(){
    int *p = (int *)malloc(sizeof(int));
    p = NULL;
    free(p);
}
```

■ What is the problem with the code?

- Compiler Error: free can't be applied on NULL pointer
- Dangling Pointer
- The program may crash as free() is called for NULL pointer.
- Memory Leak

5

Pointers

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int *ptr;
    *ptr = 10;
    *ptr = 20;
    printf("%d\n", *ptr);
    return 0;
}
```

■ What is the output?

- 10
- 20
- Segmentation fault
- Nothing of the mentioned

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int *ptr1, *ptr2;
    ptr1 = malloc(4);
    *ptr1 = 10;
    *ptr2 = free(ptr1);
    printf("%d\n", *ptr2);
    return 0;
}
```

■ What is the output?

- 10
- The address stored in ptr1
- The address stored in ptr2
- Error

L3Info – Unix/C

6

Structures

```
struct Point{
    int x = 0;
    int y = 0;
};
```

■ What is the output?

- x=0, y=0
- compilation error
- Nothing of the mentioned

```
struct student{
    char *name;
};
```

```
struct student s;
```

```
struct student fun(void){
    s.name = "newton";
    printf("%s\n", s.name);
    s.name = "alan";
    return s;
}
```

■ What is the output?

- newton alan alan
- alan newton alan
- newton alan turing
- Compilation error

```
void main(){
    struct student m = fun();
    printf("%s\n", m.name);
    m.name = "turing";
    printf("%s\n", s.name);
}
```

7

7

Problem

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

```
int *fun()
```

```
{
    int x = 5;
    return &x;
}
```

```
int main()
```

```
{
    int *p = fun();

    printf("%d", *p);
    return 0;
}
```

■ What is the problem with the code?

- Memory Leak
- Dangling pointer
- There is no problem
- compilation error

L3R&I-ARCSYS/L3Info-ARC2

8

8

Structures

```
struct point{
    int x;
    int y;
};
void foo(struct point*);
int main(){
    struct point p1 = {1, 2};
    foo(&p1);
}
void foo(struct point *p) {
    printf("%d\n", *p.x++);
}
```

■ What is the output?

- Compile time error
- Segmentation fault
- 2
- 1

```
struct point{
    int x;
    int y;
};
void foo(struct point*);
int main(){
    struct point p1 = {1, 2};
    foo(&p1);
}
void foo(struct point *p) {
    printf("%d\n", *p->x++);
}
```

■ What is the output?

- Compile time error
- Segmentation fault
- 2
- 1

9