Computer Programming 1 Lab

2020-11-19

Outline

- Pointer
- recursive function
- Exercise 7

Pointer

Pointer

- **Array**

```c
int arr[10];

printf("%d", arr[5]);
printf("%d", *(arr+5));
```

Pointer

- **2D Array**

```
int arr[10][10];

printf("%d", arr[2][3]);
printf("%d", *(*(arr+2)+3));
```

Pointer

malloc

Pointer

malloc

- □□□ #include <stdlib.h>
- □□□□

  □□□□ *Ptr;*

  Ptr = *(□□□□)*malloc(sizeof(□□□□) * □□)

Pointer

malloc

```
int arr[10];
int *arrPtr = (int*)malloc(sizeof(int) * 10);

printf("%d", arr[5]);
printf("%d", *(arrPtr+5));
```

Pointer

malloc

```
int *arrPtr = (int*)malloc(sizeof(int) * 10);
bool *arrPtr1 = (bool*)malloc(sizeof(bool) * 10);
short *arrPtr2 = (short*)malloc(sizeof(short) * 10);
float *arrPtr3 = (float*)malloc(sizeof(float) * 10);
double *arrPtr4 = (double*)malloc(sizeof(double) * 10);
long long *arrPtr5 = (long long*)malloc(sizeof(long long) * 10);
unsigned long long *arrPtr6 = (unsigned long long*)malloc(sizeof(unsigned long long) * 10);
```

Pointer

malloc

```c
int *arrPtr = malloc(sizeof(int) * 10);

//[illegible]
free(arrPtr);
```

Pointer

malloc

```c
int arr2D[2][3];

int **arr = (int**)malloc(sizeof(int*) * 2);
for (int i = 0; i < 2; i++)
    *(arr+i) = (int*)malloc(sizeof(int) * 3);

printf("%d", *(*(arr+1)+2));
```

Pointer

malloc

```c
int arr2D[2][3];

int **arr = (int**)malloc(sizeof(int*) * 2);
for (int i = 0; i < 2; i++)
    *(arr+i) = (int*)malloc(sizeof(int) * 3);

//□□□□□□□□□□□
for(int i = 0; i < 2; i++) {
    free(*(arr+i));
}
free(arr);
```

Recursive Function

Recursive Function

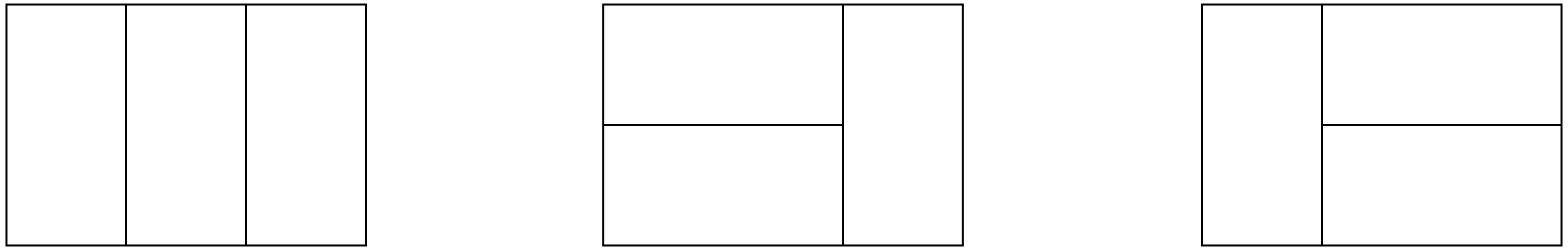
- fibonacci

```
int  fib(int n){
    if(n == 0 || n==1)
        return 1;
    else return fib(n-1) + fib(n-2);
}
```

Recursive Function

Practice

- □□□
- □□□□□1×2□□□□□2×n□□□□□□□□□



Recursive Function

Practice

- □□□
- □□□□□1×2□□□□□2×n□□□□□□□□□

```
int ans[1000]={0,1,2};
int solve(int n){
    if(n<2)
        return ans[n];
    return ans[n-1] + ans[n-2];
}
```

Recursive Function

Practice

- □□□
- □□□□□□n, k□□n^k

Recursive Function

Practice

- □□□
- □□□□□□n, k□□n^k

```
int mypow(int n, int k){
    if(k == 1)
        return n;
    else if(k%2 == 0)
        return mypow((n*n), k/2);
    else
        return (mypow((n*n), k/2)*n);
}
```

Exercise 7

Any Questions?