2021.1.24

1. 二级指针（double pointer）

1.1 二级指针储存二维数组

1. Pointer is a variable which holds the address of another variable.
2. Array is collection of consecutive location of a particular data type.

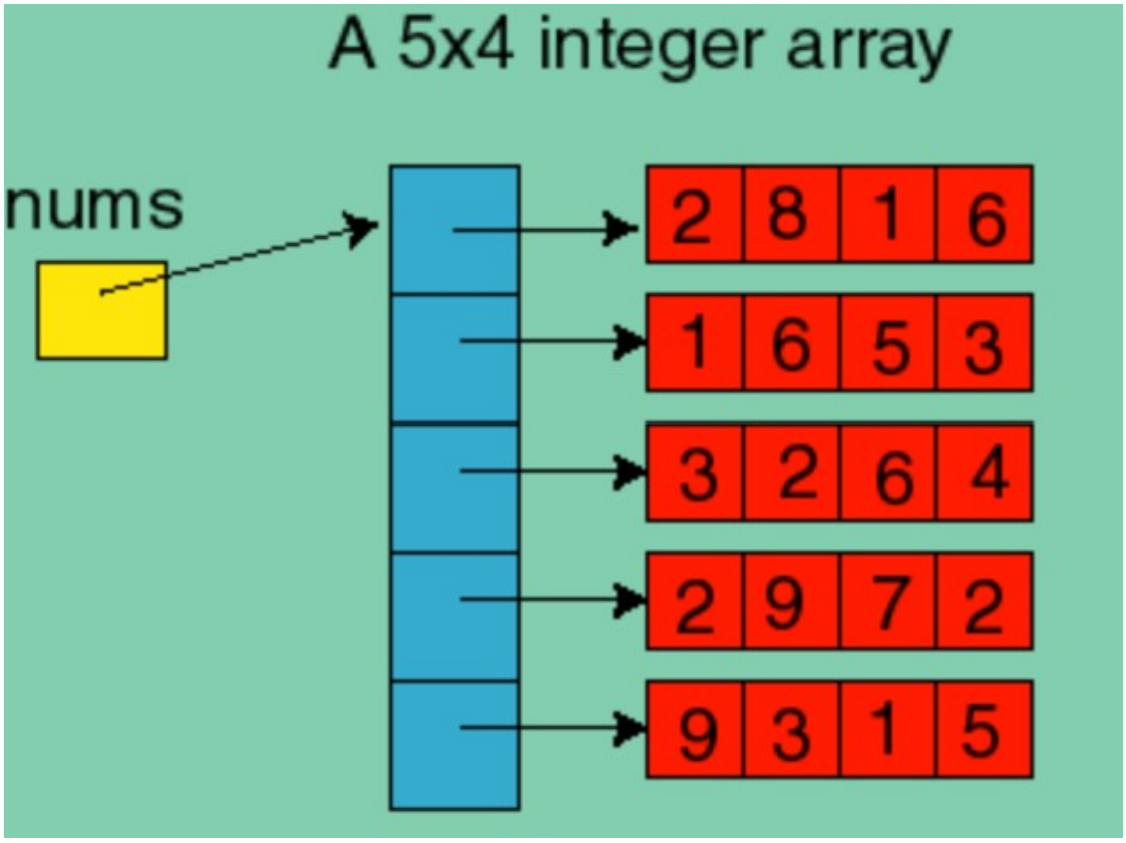
```c
#include <stdio.h>
#include <stdlib.h>


int main(){
    int **p;
    int row=5, col=4;
    // p是int的二级指针
    p = (int **)malloc(row * sizeof(int *));
    for (int i=0; i<row; i++){
        // p[i]是int的一级指针
        p[i] = (int *)malloc(col * sizeof(int *));
    }
    for (int i=0; i<row; i++){
        for (int j=0; j<col; j++){
            // p[i][j]是int整型
            p[i][j] = i+j;
            printf("%5d", p[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Output:

```
0    1    2    3
1    2    3    4
2    3    4    5
3    4    5    6
4    5    6    7
```

代码实现示意图：



1.2 二级指针与字符串/字符数组

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

void modify(char **double_pointer){
    // define a char pointer
    char *pointer = (char *)malloc(5 * sizeof(char));
    // strcpy(数组名/指针名, "test")
    strcpy(pointer, "test");
    printf("%s\n", pointer);

    // point double_pointer to pointer
    *double_pointer = pointer;
    printf("%s\n", *double_pointer);
}

int main(){
    // pointer to pointer
    char *pointer = NULL;
    modify(&pointer);
    printf("%d\n", 1);
    printf("%s\n", pointer);
}
```

注意：char *str;

- scanf("%s", str)，而不是scanf("%s", &str)
- printf("%s", str)，而不是printf("%s", *str)

2. 指针与数组

1. 数组是由地址上连续的若干个相同类型的数据组合而成的
2. 数组名<--->数组的首元素的地址，e.g. a == &a[0]
    - a+i == &a[i]
    - *(a+i) == a[i]
3. 数组中，指针的加减法计算的是类型的**偏移量**

2.1 例一

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

int main(){
    int *a;
    int num=3;
    a = (int *)malloc(num * sizeof(int));
    for (int i=0; i<num; i++){
        scanf("%d", a+i);
    }
    for (int i=0; i<num; i++){
        printf("%d\n", *(a+i));
    }
    free(a);
    return 0;
}
```

2.2 例二

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

int main(){
    int a[3] = {1, 2, 3};
    for (int *p=a; p < a+3; p++){
        printf("%d\n", *p);
    }
    return 0;
}
```

2.3 例三

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

int main(){
    int a[5] = {1, 2, 3, 4, 5};
    int *p = a;
    int *q = &a[3];
    printf("p = %p\n", p);
    printf("q = %p\n", q);
    printf("q - p = %ld\n", q - p);
    return 0;
}
```

Output:

```
p = 0x7ffeed963730
q = 0x7ffeed96373c
q - p = 3
```

2021.1.25

1. 字符串/字符数组的输入输出

1.1 scanf输入，printf输出

scanf对字符类型有%c和%s两种格式（printf同理）

1. %c用来输入单个字符
2. %s用来输入一个字符串并存在字符数组里。

%c能够识别空格和换行并将其输入，%s通过空格和换行识别一个字符串的结束。

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define maxn 256

int main(){
    char *str;
    str = (char *)malloc(maxn * sizeof(char));
    scanf("%s", str);
    printf("%s\n", str);
    return 0;
}
```

1.2 getchar输入，putchar输出

getchar和putchar分别用来输入和输出单个字符

```c
#include <stdio.h>

int main(){
    char str[5][5];
    for (int i=0; i<3; i++){
        for (int j=0; j<3; j++){
            str[i][j] = getchar();
        }
        // 这句是为了把输入中每行末尾的换行符吸收掉
        getchar();
    }
    for (int i=0; i<3; i++){
        for (int j=0; j<3; j++){
            putchar(str[i][j]);
        }
        putchar('\n');
    }
    return 0;
}
```

Input:

```
^_^
 _
 ^_
_ _
^_^
 _
```

Output:

```
^_^
 _
 ^_
_ _
^_^
 _
```

- 什么时候要吸收回车?

当按下回车，后面又是接收**字符**的scanf或getchar，由于要读取缓冲区里的内容，就会把回车取到。

但是，读取%d和%s则不会存在上述问题，因为会自动过滤空格和回车。

1.3 gets输入，puts输出

`gets()`用来输入一行字符串（注意：gets识别换行符\n作为输入结），因此scanf完一个整数，如果要使用`gets()`，需要先用`getchar()`接收正数后的换行符）

`puts()`用来输出一行字符串，并紧跟一个换行。

注意：

`gets()`后按下回车，并不会在缓冲区内留下换行符。而`scanf()`和`getchr()`后的空格和回车则会留在缓冲区内。

```

```