1、冒泡排序

1.1 冒泡排序的思想

```
1       比较相邻两个元素之间的大小，根据要求进行数据的交换。
2       排序方式：从大到小或者从小到大。
3
```

1.2 冒泡排序实现的思路

```
1  int arr[5] = {10, 40, 35, 60, 56};
2  对以上整型数组中的所有成员采用升序的方式进行排序。
3
4  第一趟：(比较4次)
5       先比较第0个元素和第1个元素的大小，如果成立则交换，
6       如果不成立则不交换，继续向下比较。
7       再比较第1个元素和第2个元素的大小，如果成立则交换，
8       如果不成立则不交换，继续向下比较。
9       再比较第2个元素和第3个元素的大小，如果成立则交换，
10      如果不成立则不交换，继续向下比较。
11      在比较第3个元素和第4个元素的大小，如果成立则交换，
12      如果不成立则不交换，继续向下比较。
13      比较完成之后，可以将最大的数放到数组的最后一个成员中。
14 第二趟：(比较3次)
15      先比较第0个元素和第1个元素的大小，如果成立则交换，
16      如果不成立则不交换，继续向下比较。
```

```
17      再比较第1个元素和第2个元素的大小，如果成立则交换，
18      如果不成立则不交换，继续向下比较。
19      再比较第2个元素和第3个元素的大小，如果成立则交换，
20      如果不成立则不交换，继续向下比较。
21      比较完成之后，可以将第二大的数放到数组的倒数第二个成员中。
22  第三趟：（比较2次）
23      先比较第0个元素和第1个元素的大小，如果成立则交换，
24      如果不成立则不交换，继续向下比较。
25      再比较第1个元素和第2个元素的大小，如果成立则交换，
26      如果不成立则不交换，继续向下比较。
27      比较完成之后，可以将第三大的数放到数组的倒数第3个成员中。
28  第四趟：（1次）
29      先比较第0个元素和第1个元素的大小，如果成立则交换，
30      如果不成立则不交换，继续向下比较。
31      比较完成之后，可以将第四大的数放到数组的倒数第4个成员中。
32
33  冒泡排序的动图链接：https://img-blog.csdnimg.cn/20210711220009741.gif
```

1.3 实现冒泡排序的代码

```
1  定义一个int类型的数组，数组成员个数为10个，
```

```
通过从终端输入的方式对数组中的成员进行初始化，
然后使用冒泡排序的方式对数组中的成员进行初始化。
封装函数：
    数组成员初始化的函数
    冒泡排序的函数
    打印数组的函数

xxx// 1. C代码的框架
// 2. 定义一个整型数组，数组成员有10个，默认初始化为0
// 3. 封装函数，通过函数完成对数组成员初始化
// 4. 封装函数，通过函数打印数组的每个元素的值
// 5. 冒泡排序的函数，实现数组中成员的排序
// 6. 在main函数中调用对应的函数进行测试

#include <stdio.h>
#define  ASC   0
#define  DESC  1

void array_init(int arr[], int length)
{
    // 数组一旦被定义，只能单个成员进行赋值
    for (int i = 0; i < length; i++)
    {
        scanf("%d", &arr[i]);
    }
}

void print_array(int arr[], int length)
{
```

```
31      for (int i = 0; i < length; i++)
32      {
33          printf("%d ", arr[i]);
34      }
35      putchar('\n');
36  }
37  // ascDesc = 0 :升序   ascDesc = 1 : 降序
38  void bubble_sort(int arr[], int length, unsigned int ascDesc)
39  {
40      int i, j;
41      for (i = 0; i < length - 1; i++)    // 循环躺数
42      {
43          for (j = 0; j < length - 1 - i; j++)    // 每趟循环的次数
44          {
45              if (ascDesc == 0)
46              {
47                  if (arr[j] > arr[j+1])    // 判断两个相邻数的大小，是否进行交换
48                  {
49                      int tmp;
50                      tmp = arr[j];
51                      arr[j] = arr[j+1];
52                      arr[j+1] = tmp;
53                  }
54              }
55              else
56              {
```

```
57                  if (arr[j] < arr[j+1])
  // 判断两个相邻数的大小，是否进行交换
58                  {
59                      int tmp;
60                      tmp = arr[j];
61                      arr[j] = arr[j+1];
62                      arr[j+1] = tmp;
63                  }
64
65              }
66
67          }
68      }
69  }
70
71  int main(int argc, const char *argv[])
72  {
73      int arr[10] = {0};
74      int len = sizeof(arr) /
  sizeof(int);
75      array_init(arr, len);
76
77      printf("排序之前 :  > ");
78      print_array(arr, len);
79
80      bubble_sort(arr, len, 0);        //
  升序
81
82      printf("升序之后 :  > ");
83      print_array(arr, len);
84
```

```
85      bubble_sort(arr, len, DESC);
    // 降序
86
87      printf("降序之后 :  > ");
88      print_array(arr, len);
89
90      return 0;
91  }
92
```

2、字符数组和字符串之间的关联

2.1 一维字符数组

```
1  一维字符数组：特定：数组中的每个成员都是一个字符类型。
```

```
 1  #include <stdio.h>
 2  #include <string.h>
 3  int main(int argc, const char *argv[])
 4  {
 5      // 1. 定义字符数组并进行初始化，
 6      // 初始化的方式和整型数组一样只是使用字符类型进行初始化
 7      char c_arr[5] = {'h', 'e', 'l', 'l', 'o'};
```

```c
    // 如果使用以上初始化的方式不可以使用%s进行打印,
    // 使用%s打印的前提，字符串的结尾为'\0'.
    // 只能使用for循环的方式一个字符一个字符的方式进行打印。
    for (int i = 0; i < sizeof(c_arr)/sizeof(char); i++)
    {
        printf("%c", c_arr[i]);
    }
    putchar('\n');

    // 使用字符对应的ascii码值进行初始化
    char c_arr1[5] = {97,98,99,100,101};
    for (int i = 0; i < sizeof(c_arr1)/sizeof(char); i++)
    {
        printf("%c", c_arr1[i]);
        //printf("%d ", c_arr1[i]);
    }
    putchar('\n');

    // 2. 使用字符数组存储字符串
    // 2.1 使用单个字符的方式进行初始化，结尾补'\0' = 0
    char c_arr2[6] = {'w', 'o', 'r', 'l', 'd', '\0'}; // 手动补尾0
    // char c_arr2[6] = {'w', 'o', 'r', 'l', 'd'}; // 自动补尾0
    // 如果使用字符数组存储字符时，有尾'\0',
```

```
31      // 可以使用%s打印字符数组中的所有的成员，不会出现越界，
32      // 打印字符串判断的是'\0'就代表结束
33      printf("%s\n", c_arr2);  // 数组名就表示字符数组的首地址
34
35
36      // 2.2. 使用字符串的表示方式对字符数组进行初始化。
37      // 要求字符数组的长度必须大于字符串的长度
38      // 字符串的长度不包含'\0'，但是存储字符串时需要存储尾'\0'
39      char c_arr3[20] = "hello world";  // 字符串长度为11，数组长度为20，可以存储此字符串
40      printf("%s\n", c_arr3);
41
42      // char c_arr4[11] = "hello world"; // 错误，字符数组越界
43      // printf("%s\n", c_arr4);
44
45      // 2.3. 先定义字符数组，然后在对字符数组进行初始化
46      char name[20] = {0};
47      scanf("%s", name);
48      printf("name = %s\n", name);
49
50  //  name = "zhoukai"; // 错误
51                        // 将字符串常量的地址赋值给name数组名，
```

```c
                        // 而数组名是一个常量，不可以进行赋值的操作
    // 如果定义字符数组完成之后，使用字符串进行赋值时，
    // 需要使用strcpy函数进行字符串的拷贝
    // #include <string.h>
    // strcpy(char *desc, const char *src);
    // 功能：字符串拷贝的函数
    // 参数：
    //      desc：  目的字符串的地址
    //      src ： 源字符串的地址
    //

    strcpy(name, "zhangsan"); // 将"zhangsan"字符串拷贝赋值给name字符数组
    printf("name = %s\n", name);
    return 0;
}

```

```
练习题1：
    定义字符数组，并使用字符串进行初始化，要求将字符串进行倒置。
    //1. C代码的框架
    //2. 定义字符数组，并进行初始化
        char c_arr[20] = "helloworld";
    //3. 打印字符串的函数
        void print_char_array(char c_arr[])
        {
```

```
            // 通过%s进行打印字符，通过'\0'判断是否结束

        }
    //4. 封装倒置的函数
        void flashback(char c_arr[])
        {
            // 通过'\0'判断字符串的长度，对字符数组进行遍历
            for (i = 0; i < len / 2 ;i++)
            {

            }
        }
-----------------------------
#include <stdio.h>
void print_char_arr(char c_arr[])
{
    printf("%s\n", c_arr);
}
// 计算字符串长度函数，
// 参数：字符串的首地址，数组名
// 返回：计算的字符串的长度，不包含'\0'
int my_strlen(char c_arr[])
{
    // 通过'\0'计算字符串的长度
    int len = 0;
#if 1
    while (c_arr[len] != '\0')
    {
```

```
37          len++;
38      }   // 退出循环之后，len中的值为字符串的
    长度
39  #else
40      int i;
41      for (i = 0; c_arr[i] != '\0';i++)
42          ;
43      len = i;
44  #endif
45
46      return len;
47  }
48
49  void char_arr_falshback(char c_arr[])
50  {
51      int len = my_strlen(c_arr);
52
53      for (int i = 0; i < len / 2 ;i++)
54      {
55          char tmp;
56          tmp = c_arr[i];
57          c_arr[i] = c_arr[len - 1 - i];
58          c_arr[len - 1 - i] = tmp;
59      }
60  }
61
62
63  int main(int argc, const char *argv[])
64  {
65      char c_arr[20] = "hello world";
66
```

```
    char_arr_falshback(c_arr);
    print_char_arr(c_arr);
    return 0;
}

```

```
练习题2:
    定义一个字符数组，char c_arr[1024]={0},
    从终端输入一个字符串对字符数组进行初始化，字符串中只能包含小写字母，
    统计字符数组中出现每个字符的个数。

    int num[26] = {0};

    num[字符-97] = num[字符-97] + 1;
#include <stdio.h>
// 字符数组初始化函数
void char_arr_init(char c_arr[])
{
    scanf("%s", c_arr);
}
// 字符数组打印的函数
void char_arr_print(char c_arr[])
{
    printf("%s\n", c_arr);
}
// 统计字符串长度的函数
int my_strlen(char c_arr[])
{
    // 通过'\0'计算字符串的长度
```

```
24      int len = 0;
25      while (c_arr[len] != '\0')
26      {
27          len++;
28      }   // 退出循环之后，len中的值为字符串的
    长度
29
30      return len;
31  }
32  // 统计字符串数组中每个字符的个数
33  void char_number(char c_arr[], int
    num[], int length)
34  {
35      int len = my_strlen(c_arr);
36      for (int i = 0; i < len; i++)
37      {
38          num[c_arr[i] - 97] =
    num[c_arr[i] - 97] + 1;
39      }
40  }
41
42  // 打印字符数组中每个字符的个数之和
43  void print_char_number(int num[], int
    length)
44  {
45      for (int i = 0 ; i < length; i++)
46      {
47          printf("%c字符的个数 = %d\n",
    i+97, num[i]);
48      }
49  }
```

```
50
51 int main(int argc, const char *argv[])
52 {
53     char c_arr[1024] = {0};
54     int num[26] = {0};
55
56     char_arr_init(c_arr);
57
58     char_number(c_arr, num, 26);
59
60     print_char_number(num, 26);
61
62
63     return 0;
64 }
65
```

2.2 string.h头文件中提供的字符串处理函数

```
1 https://www.runoob.com/cprogramming/c-
  standard-library-string-h.html
```

2.2.1 strlen函数

```
#include <string.h>

size_t strlen(const char *s);

功能：计算字符串的长度，不包含尾'\0'
参数：
    s ： 字符串的首地址/字符数组的数组名
返回值：
    计算的字符串的长度
```

```c
#include <stdio.h>
#include <string.h>
int main(int argc, const char *argv[])
{
    char arr[20] = "hello world";

    printf("字符串的长度 = %ld\n", strlen(arr));
    printf("字符数组的长度 = %ld\n", sizeof(arr));

    // 注：求字符串的长度就用strlen函数，千万不要使用sizeof

    return 0;
}

```

2.2.2 strcpy函数

```
#include <string.h>

char *strcpy(char *dest, const char *src);
功能：字符串拷贝的函数
参数：
    @ dest:拷贝到缓冲区的首地址/字符数组的数组名
    @ src:被拷贝字符串的首地址/字符数组的数组名
返回值：
    返回：拷贝到目标地址空间的首地址

char *strncpy(char *dest, const char *src, size_t n);
功能：字符串拷贝的函数，拷贝n个字符
参数：
    @ dest:拷贝到缓冲区的首地址/字符数组的数组名
    @ src:被拷贝字符串的首地址/字符数组的数组名
    @ n : 拷贝字符的个数
返回值：
    返回：拷贝到目标地址空间的首地址
```

```c
#include <stdio.h>
#include <string.h>
int main(int argc, const char *argv[])
```

```c
{
    char arr1[20] = "hello";
    char arr2[20] = "world";
    char arr3[20] = {0};

    // 使用strcpy实现字符串的拷贝，也会将尾'\0'进行拷贝
    printf("拷贝之前arr1 = %s\n", arr1);
    strcpy(arr1, arr2); // 注意数组越界的问题
    printf("拷贝之后arr1 = %s\n", arr1);


    // 注：一定定义好字符数组之后，不可以对字符数组使用字符串赋值，
    // 需要使用字符串拷贝的函数进行字符串的拷贝赋值。
    char name[20] = {0};
    // name = "lisi";  // error
    strcpy(name, "lisi");  // 字符串的拷贝赋值

    // 使用strncpy实现字符串的拷贝
    printf("拷贝之前arr3 = %s\n", arr3);
    // 拷贝字符个数完成之后，不会拷贝尾'\0'，也不会自动添加'\0'
    strncpy(arr3,arr2,3);
    arr3[4] = '\0'; // 手动添加一个尾'\0'
    printf("拷贝之后arr3 = %s\n", arr3);


```

```
27
28      return 0;
29  }
30
```

2.2.3 strcat函数

```
1  #include <string.h>
2
3  char *strcat(char *dest, const char
   *src);
4  功能：字符串的拼接
5  参数：
6       @ dest:拷贝到缓冲区的首地址/字符数组的数
   组名
7       @ src:被拷贝字符串的首地址/字符数组的数组
   名
8       将第二个参数对应的字符串拼接到第一个字符串
   的结尾。
9  返回值：
10      返回：拷贝到目标地址空间的首地址
11 char *strncat(char *dest, const char
   *src, size_t n);
12      功能：字符串的拼接，  拷贝n个字符
13 参数：
14      @ dest:拷贝到缓冲区的首地址/字符数组的数
   组名
15      @ src:被拷贝字符串的首地址/字符数组的数组
   名
16      @ n ： 拷贝字符的个数
```

```
17      将第二个参数对应的字符串拼接到第一个字符串的结尾。
18  返回值:
19      返回：拷贝到目标地址空间的首地址
```

```c
#include <stdio.h>
#include <string.h>
int main(int argc, const char *argv[])
{
    char name[20] = {0};
    char firstName[10] = "zhou";
    char lastName[10] = "kai";

    strcpy(name, firstName);

    // 将第二个字符串拼接到第一个字符串的结尾
    strcat(name, lastName);
    printf("拼接之后 name = %s\n", name);


    char sex[10] = " MAN";
    strncat(name, sex, 2);
    printf("拼接之后 name = %s\n", name);

    return 0;
}

```

2.2.4 strcmp函数

```
#include <string.h>

int strcmp(const char *s1, const char *s2);
功能：比较两个字符串的大小
参数：
    @ s1 : 第一个字符串的首地址
    @ s2 : 第二个字符串的首地址
返回值：
    s1 == s2 : 返回0
    s1 > s2 : 返回大于0的数
    s1 < s2 : 返回小于0的数

int strncmp(const char *s1, const char *s2, size_t n);
功能：比较两个字符串的大小，比较前n个字符的大小
参数：
    @ s1 : 第一个字符串的首地址
    @ s2 : 第二个字符串的首地址
    @ n : 比较n个字符的大小
返回值：
    s1 == s2 : 返回0
    s1 > s2 : 返回大于0的数
    s1 < s2 : 返回小于0的数
```

```
#include <stdio.h>
#include <string.h>
int main(int argc, const char *argv[])
```

```c
{
    char s1[20] = "hello";
    char s2[20] = "world";

    if (strcmp(s1, s2) == 0)
    {
        printf("s1 == s2\n");
    }
    else if (strcmp(s1, s2) > 0)
    {
        printf("s1 > s2\n");
    }
    else
    {
        printf("s1 < s2\n");
    }

    char name1[20] = "lisi";
    char name2[20] = "liyi";
    if (strncmp(name1, name2, 2) == 0)
    {
        printf("前两个字符相等\n");
    }
    else
    {
        printf("前两个字符不相等\n");
    }


    return 0;
}
```

```
35
```

```
作业：
    1. 实现my_strlen函数
        判断字符串的尾'\0'
    2. 实现my_strcpy函数
        将第二个尾'\0'之前的所有的字符一个一个的进行拷贝，
        包括尾'\0'也要进行拷贝，
        到第二个字符串的尾'0'之后，拷贝结束。
    3. 实现my_strcat函数
        判读第一个字符的结尾，然后将第二个字符串的一个一个字符的
        拷贝到第一个字符的结尾之后。
    4. 实现my_strcmp函数
        依次比较每个字符的大小，如果相等比较下一个，
        如果不相等，返回两个字符进行相减的值，
        不相等时循环结束，尾'0'结束，字符串相等结束。
```

2.3 二维字符数组

```
二维字符数组：特定：数组中的每个成员都是一个字符类型。
```

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

```c
void char_arr_print(char s1[][5], int row, int col)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            putchar(s1[i][j]);
        }
        putchar('\n');
    }
}

void char_arr_show(char s[][6], int row)
{
    for (int i = 0; i < row; i++)
    {
        // s[0] : 表示第0行的首地址
        // s[1] : 表示第1行的收地址
        // 。。。。。。
        printf("%s\n", s[i]);
    }
}

int main(int argc, const char *argv[])
{
    // 1. 定义字符数组，单独对每个元素初始化
    char s1[2][5] = {{'h','e','l','l','o'},
    {'w','o','r','l','d'}};
```

```
30      // 如果没有尾0只能使用for循环嵌套的方式，一个字符一个字符的打印
31      char_arr_print(s1, 2, 5);
32
33
34      // 2. 使用字符串的方式对二维字符数组的每一行进行初始化
35      // 要求二维字符数组的每一列字符的个数要大于字符串的长度
36      //char s2[2][5] = {"hello", "world"};        // 错误，越界
37
38      char s2[2][6] = {"hello", "world"};  // 常用
39      char_arr_show(s2, 2);
40
41      char s3[2][6] = {{"zhang"}, {"lisi"}};
42      char_arr_show(s3, 2);
43
44      // 3. 先定义二维数组，后对二维数组进行初始化，
45      // 不可以使用直接赋值的方式，需要使用字符串拷贝的方式
46      char s4[2][6] = {0};
47      strcpy(s4[0], "zhou");
48      strcpy(s4[1], "dai");
49      char_arr_show(s4, 2);
50
51      return 0;
52  }
```

```
53
```

```
练习题：
    定义一个二维字符数组，并对其进行初始化，
    使用冒泡排序的方式，对字符串进行排序。
    strcmp
    strcpy

    char name[5][10] = {"zhangsan","lisi","busan","busi","wangwu"};

    char tmp[10];
    strcpy(tmp, name[0]);
    strcpy(name[0], name[1]);
    strcpy(name[1], tmp;)
```

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

void arr_string_print(char s[][10], int row)
{
    for (int i = 0; i < row; i++)
    {
        printf("%s ", s[i]);
    }
    putchar('\n');
}

```

```c
void string_bubble_sort(char s[][10], int row)
{
    for (int i = 0; i < row - 1; i++)
    {
        for (int j = 0; j < row - 1 - i; j++)
        {
            if (strcmp(s[j], s[j+1]) > 0)
            {
                char tmp[10] = {0};
                strcpy(tmp, s[j]);
                strcpy(s[j], s[j+1]);
                strcpy(s[j+1], tmp);
            }
        }
    }
}

int main(int argc, const char *argv[])
{
    char name[5][10] = {"zhangsan","lisi","busan","busi","wangwu"};

    // 排序之前
    arr_string_print(name, 5);
    string_bubble_sort(name, 5);
    // 排序之后
    arr_string_print(name, 5);
```

```
39
40
41      return 0;
42  }
43
```

3、指针

1.1 指针的概念

```
1   内存访问的最小单位为字节，1字节是8bit位。
2   内存的访问本质是通过地址进行访问的，每个字节都有要给唯一的地址。
3   内存的地址是连续的。
4
5   地址可以理解为指针。
6
7   专门用来存储指针(地址)的变量，称为指针变量，
8   要想使用一个变量存储内存的地址，因此需要定义指针类型的变量。
9
10  定义指针变量和定义普通变量的区别：
11  普通变量就是存储的普通的数据，比如整型数据，字符数据。
12  指针变量就是用来存储内存地址的变量，因此要想使用一个变量存储内存的地址，
13      需要定义指针类型的变量。
14
15      地址：  访问内存的一个编号
```

```
16      指针：  用来存储地址的指针变量
17
18  定义指针：就是再说定义指针类型的变量。
19
20  指针：地址
21  指针：指针变量
22  指针：数据类型(指针类型)
```

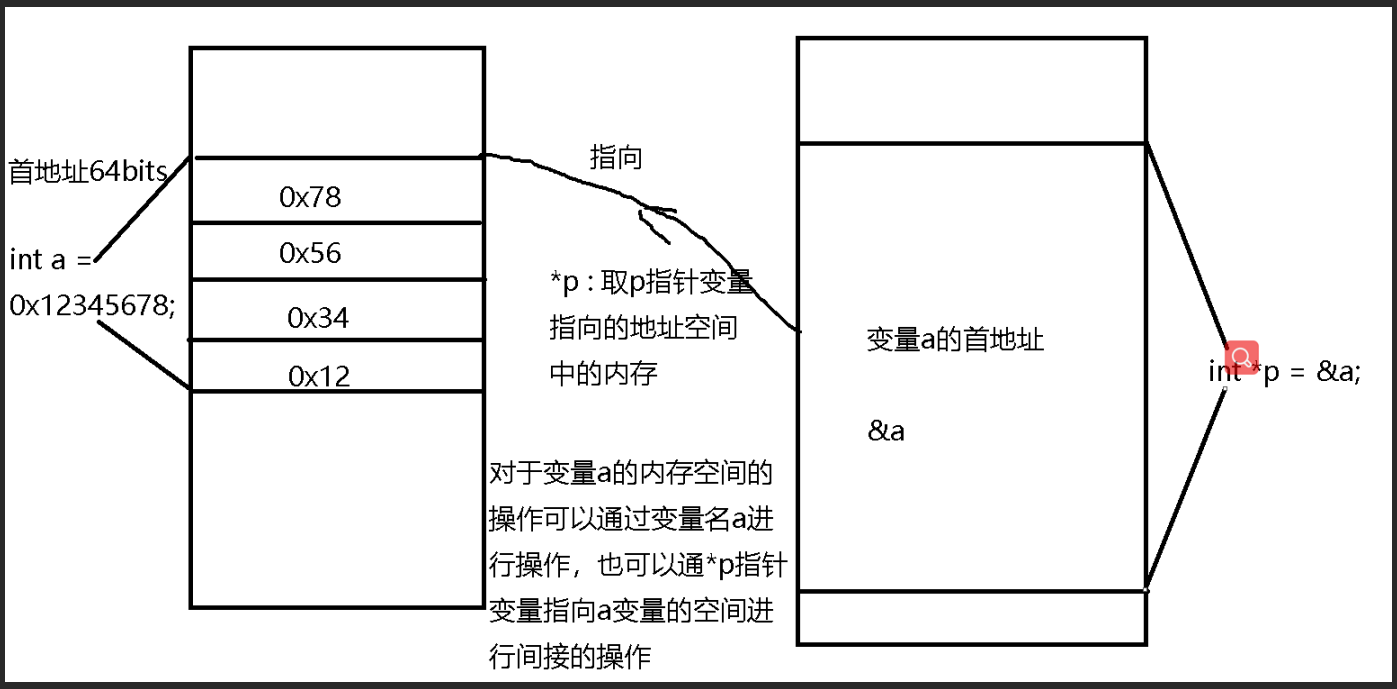
1.2 定义指针类型的变量及初始化

```
1  定义指针变量的格式：
2      存储类型  数据类型   *指针变量名;
3                        |     |       |----> 指针变量名，需要使用地址进行初始化
4                        |     |----> 定义指针类型的变量
5                        |---> 数据类型(基本类型/构造类型)
6      存储类型  数据类型*   指针变量名;
7                        |    |       |----> 指针变量名，需要使用地址进行初始化
8                        |    |----> 定义指针类型的变量
9                        |---> 数据类型(基本类型/构造类型)
10     存储类型  数据类型   *   指针变量名;
11                       |     |       |----> 指针变量名，需要使用地址进行初始化
```

```
                    |     |----> 定义指针类型的变量
                    |---> 数据类型(基本类型/构造类型)
指针变量的初始化:
    1> 定义指针变量的同时进行初始化
        int a = 100;
        int *p = &a;
    2> 先定义指针变量，后进行初始化
        int a = 200;
        int *p = NULL;  // NULL : 0地址
                    // 防止野指针的出现，如果定义指针变量时，没有初始化，
                    // 指针变量中存储的就是一个随机的地址，访问此地址，
                    // 就会访问非法的内存空间，导致段错误的出现，
                    // 野指针:  指针变量指向的地址空间值不确定。
        p = &a;

*和&的作用: 都属于单目运算符
    & :  放到变量名的前边表示对变量进行取地址运算。
    * :
        1> 定义变量时，放到数据类型和指针变量名之间，表示定义指针类型的变量。
            int *p = &a;
        2> 放到指针变量名前，表示取指针变量指向的空间的内容。
```

```
33                  int b = *p; // 将指针变量指向的内存空间的内容赋值给变量b。
```




```
明天授课内容：
    指针

作业：
    将今天的一维字符数组，二维字符数组，string.h相关的函数理解

    实现strlen,strcpy,strcat,strcmp实现。

    char *strcpy(char dest[], char src[])
    {

        return dest;
    }
```