指针的定义的定义与使用

2023年9月10日 22:40

7 指针

7.1 指针的基本概念

指针的作用就是访问不同区的数据

指针的作用： 可以通过指针间接访问内存

- 内存编号是从0开始记录的，一般用十六进制数字表示
- 可以利用指针变量保存地址

7.2 指针变量的定义和使用

指针变量定义语法： `数据类型 * 变量名;`

示例：

```
int main() {

    //1、指针的定义
    int a = 10; //定义整型变量a

    //指针定义语法： 数据类型 * 变量名 ;
    int * p;

    //指针变量赋值
    p = &a; //指针指向变量a的地址
    cout << &a << endl; //打印数据a的地址
    cout << p << endl;  //打印指针变量p

    //2、指针的使用
    //通过*操作指针变量指向的内存
    cout << "*p = " << *p << endl;

    system("pause");

    return 0;
}
```

指针所占内存空间

2023年9月14日 11:06

用sizeof语句即可，32位操作系统为4，64位操作系统为8

空指针与野指针

2023年9月14日 21:29

7.4 空指针和野指针

空指针：指针变量指向内存中编号为0的空间

用途：初始化指针变量

注意：空指针指向的内存是不可以访问的

空指针：指针变量指向内存中编号为0的空间

用途：初始化指针变量

注意：空指针指向的内存是不可以访问的

```
int main() {

    //指针变量p指向内存地址编号为0的空间
    int * p = NULL;

    //访问空指针报错
    //内存编号0 ~255为系统占用内存，不允许用户访问
    cout << *p << endl;

    system("pause");

    return 0;
}
```

野指针：指针变量指向非法的内存空间

示例2：野指针

```
int main() {

    //指针变量p指向内存地址编号为0x1100的空间
    int * p = (int *)0x1100;

    //访问野指针报错
    cout << *p << endl;

    system("pause");

    return 0;
}
```

野指针：指针变量指向非法的内存空间

示例2：野指针

```
int main() {

    //指针变量p指向内存地址编号为0x1100的空间
    int * p = (int *)0x1100;

    //访问野指针报错
    cout << *p << endl;

    system("pause");

    return 0;
}
```

> 总结：空指针和野指针都不是我们申请的空间，因此不要访问。

const修饰指针

2023年9月14日 21:41

7.5 const修饰指针

const修饰指针有三种情况：

1. const修饰指针 --- 常量指针
2. const修饰常量 --- 指针常量
3. const即修饰指针，又修饰常量

```
int main() {

    int a = 10;
    int b = 10;

    //const修饰的是指针，指针指向可以改，指针指向的值不可以更改
    const int * p1 = &a;
    p1 = &b; //正确
    //*p1 = 100;  报错


    //const修饰的是常量，指针指向不可以改，指针指向的值可以更改
    int * const p2 = &a;
    //p2 = &b; //错误
    *p2 = 100; //正确

    //const既修饰指针又修饰常量
    const int * const p3 = &a;
    //p3 = &b; //错误
    //*p3 = 100; //错误

    system("pause");

    return 0;
```

技巧：看const右侧紧跟着的是指针还是常量，是指针就是常量指针，是常量就是指针常量

常量指针

特点：指针的指向可以修改，指针指向的值不可以改

指针常量

特点：指针的值可以改，指针的指向不可以改

指针和数组

2023年9月15日 15:11

7.6 指针和数组

作用：利用指针访问数组中元素

```
int main() {

    int arr[] = { 1,2,3,4,5,6,7,8,9,10 };

    int * p = arr;  //指向数组的指针

    cout << "第一个元素：  " << arr[0] << endl;
    cout << "指针访问第一个元素：  " << *p << endl;

    for (int i = 0; i < 10; i++)
    {
        //利用指针遍历数组
        cout << *p << endl;
        p++;
    }

    system("pause");

    return 0;
}
```

p++的理解：通过加一来增加字节（内存）使指针内存增加实现访问下一个元素

两种定义：

指针和函数

2023年9月14日 21:41

7.7 指针和函数

作用：利用指针作函数参数，可以修改实参的值

与值传递那一节来做对比，理解其意义

```
//值传递
void swap1(int a ,int b)
{
    int temp = a;
    a = b;
    b = temp;
}
//地址传递
void swap2(int * p1, int *p2)
{
    int temp = *p1;
    *p1 = *p2;
    *p2 = temp;
}

int main() {

    int a = 10;
    int b = 20;
    swap1(a, b); // 值传递不会改变实参

    swap2(&a, &b); //地址传递会改变实参

    cout << "a = " << a << endl;

    cout << "b = " << b << endl;

```

指针，数组，函数

2023年9月15日 15:11

7.8 指针、数组、函数

案例描述：封装一个函数，利用冒泡排序，实现对整型数组的升序排序

例如数组：int arr[10] = { 4,3,6,9,1,2,10,8,7,5 };

```
#include<iostream>
using namespace std;
void bubbleSort(int* arr,int len ) {  //地址传递，用指针来接收
    for (int i = 0; i < len - 1; i++) {
        for (int j = 0; j < len - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;}}}}
int main() {
    int arr[10] = { 4,6,3,1,0,9,8,7,5,2 };
    cout << "排序前：";
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << "  ";}
    cout << endl;
    int len = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, len); //arr即为arr数组的首地址（联系如何访问数组的首地址即可）；地址传递，可以修改实参的值
    cout << "排序后：";
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << "  ";}
    system("pause");
```