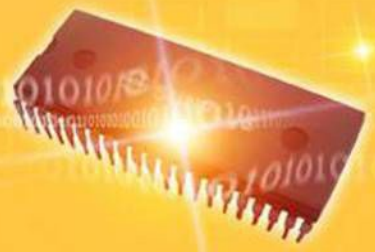
凌阳教育
www.sunplusedu.com
值得信赖的教育品牌

嵌入式系统工程师



Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

指针的概念与应用

指针

爱它，就让它去学C语言的指针

恨它，也让它去学C语言的指针

指针是C语言里面**最重要**也是**最难理解**的知识

学习了指针，才算真正踏入C语言的大门

大纲

- 有关内存的那点事
- 指针的相关概念—指针与指针变量
- 变量与指针
- 字符串与指针
- 数组与指针
- 函数与指针
- 其它特殊指针

大纲

- 有关内存的那点事
- 指针的相关概念—指针与指针变量
- 变量与指针
- 字符串与指针
- 数组与指针
- 函数与指针
- 其它特殊指针

有关内存的那点事

内存—很熟悉的名词，但又觉得很神秘

在学习指针之前先让我们了解一些内存的基本知识

有关内存的那点事

- **内存含义**
  - **存储器**：计算机的组成中，用来存储程序和数据，辅助CPU进行运算处理的重要部分
  - 内存：内部存贮器，**暂存**程序/数据——掉电丢失

    SRAM、DRAM、DDR、DDR2、DDR3
  - 外存：外部存储器，**长时间**保存程序/数据—掉电不丢失

    ROM、ERRROM、FLASH（NAND、NOR）、硬盘、光盘
- **内存的作用(内存储器)：**

  内存是沟通cpu与硬盘的桥梁：

  暂存放CPU中的运算数据

  暂存与硬盘等外部存储器交换的数据

有关内存的那点事

- 有关内存的两个概念：

  物理存储器和存储地址空间

- 物理存储器：**实际存在的具体存储器芯片**
  - 主板上装插的内存条
  - 显示卡上的显示RAM芯片
  - 各种适配卡上的RAM芯片和ROM芯片

- 存储地址空间：**对存储器编码的范围**
  - 编码：对每个物理存储单元（一个字节）分配一个号码
  - 寻址：可以根据分配的号码找到相应的存储单元，完成数据的读写
  - 我们在软件上常说的内存是指这一层含义

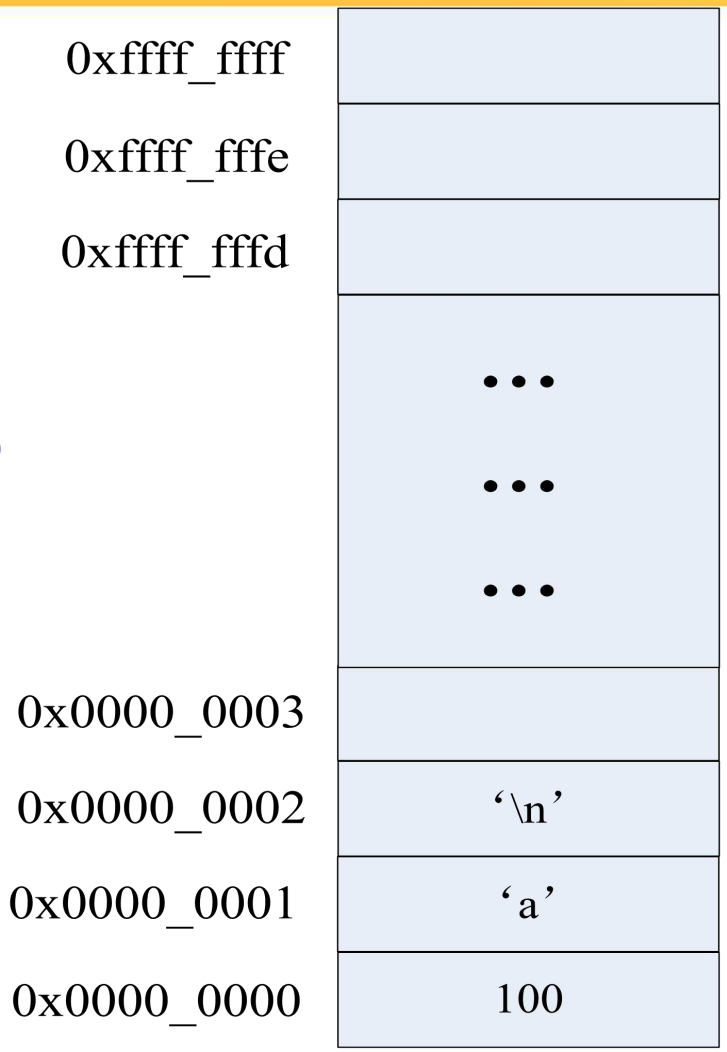
有关内存的那点事

- 我们软件上（C语言中）
  - 将内存抽象成一个很大的**一维字符数组**
  - 编码就是对内存的**每一个字节**分配一个**32位或64位**的编号（与32位或者64位处理器相关）
  - 这个编号我们称之为**内存地址**。
- 内存中的每一个数据都会分配相应的地址

  char: 占一个字节分配一个地址

  int: 占四个字节分配四个地址

  float、struct、函数、数组等



有关内存的那点事

- 无规矩不成方圆
  - 内存中的数据如果乱糟糟的存放，使用肯定不方便
  - 内存是按区域分类存放数据的

栈区(stack) 存放函数的参数值、返回值、局部变量等
堆区(heap) 用于动态内存分配
未初始化数据(bss) 全局未初始化、静态未初始化数据
初始化数据(data) 全局初始化、静态初始化数据
文字常量区(text) 字符串常量
代码区(text) 可执行文件的二进制代码(函数)

- 01.memory.c

```c
#include <stdio.h>
#include <stdlib.h>
int e;
static int f;
int g = 10;
static int h = 10;
int main(int argc, char *argv[])
{
    int a;
    int b = 10;
    static int c;
    static int d = 10;
    char *i = "test";
    char *k = NULL;

    printf("&a\t %p\t //局部未初始化变量\n",&a);
    printf("&b\t %p\t //局部初始化变量\n", &b);

    printf("&c\t %p\t //静态局部未初始化变量\n", &c);
    printf("&d\t %p\t //静态局部初始化变量\n", &d);

    printf("&e\t %p\t //全局未初始化变量\n", &e);
    printf("&f\t %p\t //全局静态未初始化变量\n", &f);

    printf("&g\t %p\t //全局初始化变量\n", &g);
    printf("&h\t %p\t //全局静态初始化变量\n", &h);

    printf("i\t %p\t //只读数据(文字常量区)\n", i);

    k = (char *)malloc(10);
    printf("k\t %p\t //动态分配的内存\n", k);
    return 0;
}
```

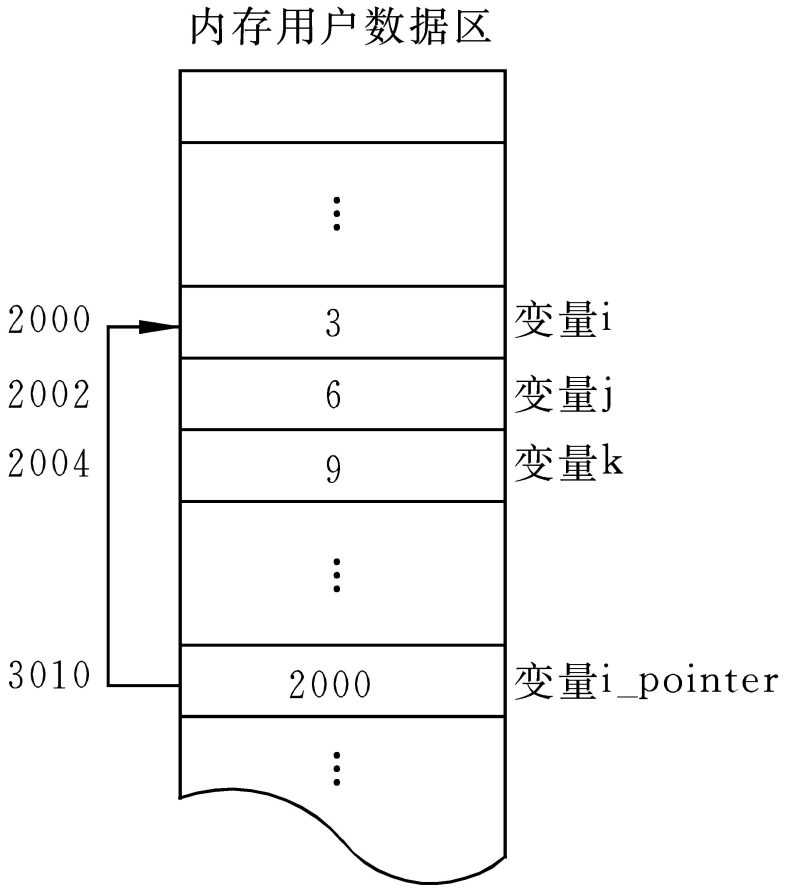
大纲

- 有关内存的那点事
- 指针的相关概念—指针与指针变量
- 变量与指针
- 字符串与指针
- 数组与指针
- 函数与指针
- 其它特殊指针

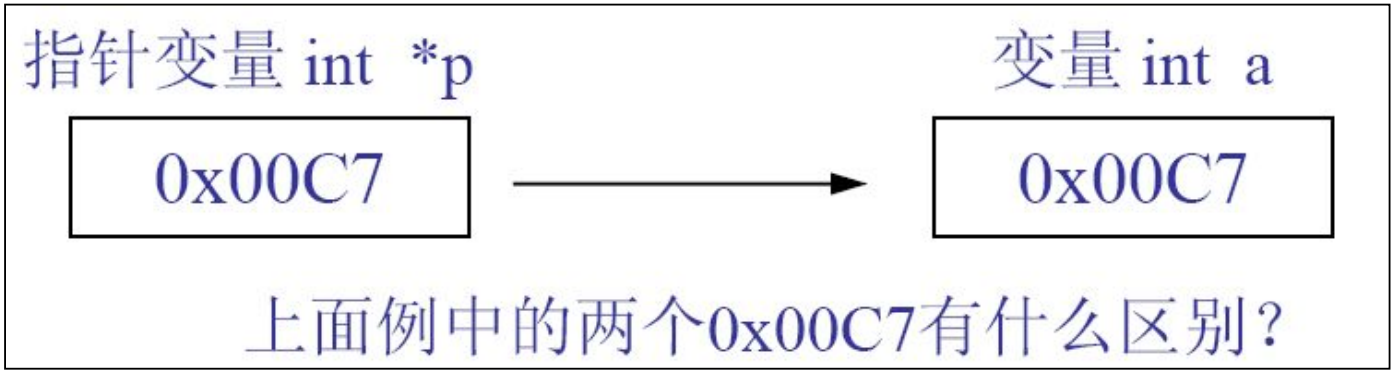
指针的相关概念

- **指针:**
  - 内存区的每一个字节都有一个编号,这就是**“地址”**
  - 如果在程序中定义了一个变量,在对程序进行编译或运行时,系统就会给这个变量分配**内存单元**,并确定它的内存**地址**(编号)
  - **指针**的实质就是内存**“地址”**

- **指针变量:**
  - 用来存放指针(地址)的变量,它的值通常是另一个变量的地址.



指针的相关概念



说明:

a: 是一个普通变量，0x00C7是一个普通的16进制数.

p: 是指针变量，0x00C7是另一个变量在内存中的地址.

指针的相关概念

➢ 指针变量实例

1. float score;
   float *p = &socre; //指向float型变量的指针变量
2. int a[5];
   int *q = a; //指向数组第0个元素的指针变量
3. char *string = “hello”; //指向字符串的指针变量
4. int (*foo)(int x, int y);//指向函数的指针变量
5. struct student *boy; //指向结构体的指针变量

大纲

- 有关内存的那点事
- 指针的相关概念—指针与指针变量
- 变量与指针
- 字符串与指针
- 数组与指针
- 函数与指针
- 其它特殊指针

指针变量的定义

- **指针变量的定义格式:**

  类型标识符 *变量名;

  如: int *point;

```
int main ( )
{
    int  a = 100, b = 200;
    int *p_1,*p_2 = &b;
    p_1 = &a;
    printf( "%d\n" , a);
    printf( "%d\n" , *p_1);
    printf( "%d\n" , b);
    printf( "%d\n" , *p_2);
    return 0;
}
```

- *表示该变量的类型是一个指针变量，指针变量名是p_1而不是*p_1.
- 指针变量可以在定义的时候初始化
- 也可以先定义再初始化.
- *p_1是p_1指向的变量的值.
- 一个指针变量只能指向同一个类型的变量.

指针变量的引用

- 对于指针变量的两个运算符:
  1. &取地址运算符
  2. *指针运算符

- 如果已执行了语句:pointer_1 = & a;
  1. &* pointer_1的含义是什么?
  2. *& a的含义是什么?
  3. (*pointer_1)++相当于a++。

指针变量的引用

02.point_var.c

```
1  #include <stdio.h>
2  int  main()
3  {
4      int *p1,*p2,temp,a,b;
5      p1=&a;
6      p2=&b;
7      printf("请输入:a b的值:\n");
8      scanf("%d %d",p1,p2);
9      if(*p1>*p2)
10     {
11         temp = *p1; *p1 = *p2; *p2 = temp;
12     }
13     printf("a=%d b=%d\n",a,b);
14     printf("*p1=%d *p2=%d\n",*p1,*p2);
15     return 0;
16 }
```

指针变量的引用

- 不同类型的指针相互赋值——强制类型转换

03.point_force.c

```
#include <stdio.h>
int  main()
{
    int a=0x1234,b=0x5678;
    char *p1,*p2;
    printf("%0x %0x\n",a,b);

    p1=(char *)&a;
    p2=(char *)&b;
    printf("%0x %0x\n",*p1,*p2);

    p1++;
    p2++;
    printf("%0x %0x\n",*p1,*p2);
    return 0;
}
```

指针变量的引用

- 指针变量的大小
- 使用sizeof()测量指针的大小，得到的总是:4或8

```
char *p1;
short int *p2;
int *p3;
struct stu *p4;
```

均为4

原因：sizeof()测的是指针变量指向存储地址的大小

在32位平台，所有的指针（地址）都是32位

在64位平台，所有的指针（地址）都是64位

大纲

- 有关内存的那点事
- 指针的相关概念—指针与指针变量
- 变量与指针
- 字符串与指针
- 数组与指针
- 函数与指针
- 其它特殊指针

字符串与指针

- 字符串的存储形式： **数组、字符串指针、堆**

  1、char string[] = “I love C!”

  2、char *str = “I love C!”

  3、char *str =(char*)malloc(10*sizeof(char));

  strcpy(str,"I love C");

- **字符数组:**

  在内存（栈、全局区）中开辟了一段空间存放字符串

- **字符串指针:**

  在文字常量区开辟了一段空间存放字符串，将字符串的**首地址**付给str

- **堆:**

  使用malloc函数在堆区申请空间，将字符串拷贝到堆区

字符串与指针

1、初始化

- 字符数组、指针指向的字符串：定义时直接初始化

```
char buf_aver[]={"hello world"};
char *buf_point="hello world";
```

- 堆中存放的字符串

不能初始化、只能使用strcpy、scanf赋值

```
char *buf_heap;
buf_heap=malloc(10);
strcpy(buf_heap,"hell world");
```

字符串与指针

2、使用时赋值

字符数组：使用scanf或者strcpy

```
buf_aver="hello kitty";                  错误
strcpy(buf_aver,"hello kitty");   正确
scanf("%s",buf_aver);                    正确
```

指向字符串的指针：

```
buf_point="hello kitty";                 正确，改指向另一个字符串
strcpy(buf_point,"hello kitty"); 错误，只读
```

3、可修改性：

- 数组和堆中的字符串：**可读可写**
  存储在栈区、全局区、堆区
- 指针指向的字符串：　**只读**
  存储在文字常量区

字符串与指针

04.string_point.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char str_array[]= "I love C_avr!";
    char *str_point = "I love C_point!";
    char *str_heap=NULL;

    str_heap = (char *)malloc(20);
    strcpy(str_heap,"I love C_heap");

    printf("%s,%s,%s\n",str_array,str_point,str_heap);
    printf("%p,%p,%p\n",str_array,str_point,str_heap);
    printf("%d,%d,%d\n",sizeof(str_array),sizeof(str_point),sizeof(str_heap));
    printf("%d,%d,%d\n",strlen(str_array),strlen(str_point),strlen(str_heap));

    str_array[0] = 'y';                 //可自由修改变量内容
    printf("%s\n",str_array);

    //*(str_point+0) = 'y';             //不可修改变量内容
    printf("%s\n",str_point);
    return 0;
}
```

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com