数据的储存

2022年2月8日 19:54

一、数据类型介绍

整形家族

- char
  - Unsigned char
  - Signed char
- short
  - Unsigned short [int]
  - Signed short [int]
- int
  - Unsigned int
  - Signed int
- long
  - Unsigned long [int]
  - Signed long [int]

浮点型家族

- float
- double

构造类型

- 数组类型
  - 例如：Int arr [5] 和 Int arr [6]并不是同一个类型
- 构造体类型 struct
- 枚举类型 enum
- 联合类型 uniom

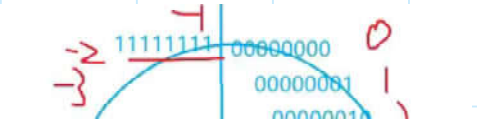
指针类型

- char*p;
- int* p;
- float* p;
- void* p;

空类型:

- void表示空类型(无类型)
- 通常应用于函数的返回类型、函数的参数、指针类型。

二、整形在内存中的储存



1、数据范围

1、数据范围

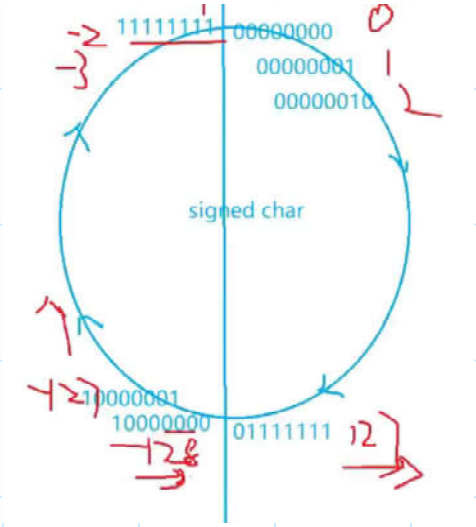
例如“char”类型，最高位用零表示正负

则正数：00000001 ~ 01111111（1~127）

零：00000000

负数：10000001 ~ 11111111 （-1~-127）

特殊：10000000 （-128）



2、原码、反码、补码

计算机中的有符号数（整数）有三种表示方法,即原码、反码和补码

cpu只有加法器

计算机的内存地址和内存中的数据用16进制表示

4个二进制数可化为1个16进制数

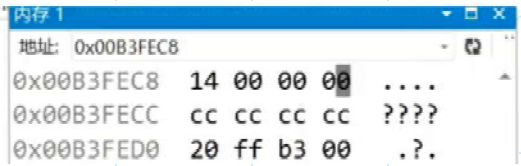
3、大端小端存储模式（多个字节）

大端(存储)模式,是指数据的低位保存在内存的高地址中,而数据的高位,保存在内存的低地址中

小端(存储)模式,是指数据的低位保存在内存的低地址中,而数据的高位,保存在内存的高地址中

```
int a = 20;
//00000000000000000000000000010100
//0x00 00 00 14
```

4



可以看出当前机器为小端存储模式

求节序例题：

```
#include<stdio.h>

int check_sys()
{
	int a = 1;
	return *(char*)&a;	//char* 只访问一个字节
}

int main()
{
	int ret = check_sys();
	if (ret == 1)
	{
		printf("小端\n");
```

取值范围与补码例题：

```
int main()
{
	char arr[1000];
	int i = 0;
	for (i = 0; i < 1000; i++)
	{
		arr[i] = -1 - i;
	}
	printf("%d", strlen(arr));

	return 0;
}
//答案为255
```

```
    }
    else
    {
        printf("大端\n");
    }

    return 0;
}
```

三、浮点数的储存

1、定义

IEEE标准下，任何浮点数都可以用以下形式表达：

符号位 + 有效数字 + 2^指数位

（原：(-1)^S * M * 2^E）

例：十进制 5.5 ,写成二进制是 101.1 ,相当于 1.011x2^2

另外IEEE规定：由于有效数字第一位总是1，所以将其省略，读取时再添上，这样可以节省一位有效数字

有效数字的储存：从高到低排（小数在后面加0不会影响值的大小）

例如3：浮点数：10100000 00000000 0000000

整数：00000000 00000000 00000000 00000101

E的值

我们知道,科学计数法中的E是可以出现负数的，所以EEE 754规定：

存入内存时E的真实值必须再加上一一个中间数

对于8位的E ,这个中间数是127

对于11位的E ,这个中间数是1023

例如, 2^10的E是10 ,所以保存成32位浮点数时,必须保存成10+127=137 ,即10001001。

E的特殊值

- E全为0
  - 符号为正：+0
  - 符号为负：-0
  - 且取出时不在前面加1，还原为0.xxxxxx
- E全为1
  - 符号位正：正无穷
  - 符号为负：负无穷