# 指针前奏

## 指针的重要性

指针是C语言中非常重要的数据类型，如果你说C语言中除了指针，其他你都学得很好，那你干脆说没学过C语言。

## 小需求

* void change(int n)函数调用完毕后，改变实参的值
* 分析：修改实参的值->找到存储空间->地址

# 指针变量的定义

## 定义的格式

* **类名标识符  \*指针变量名;**
* **int \*p**;

## 先定义后赋值

* 简单取值

int a = 10;

int \*p;

p = &a;

printf(“%d”, \*p);

* 简单改值

\*p = 9;

## 定义的同时赋值

int a = 10;

int \*p = &a;

## 实现修改实参

## 注意点

* int \*p; p = 1000;
* int \*p; \*p = 100;
* int \*p; \*p = &a;
* %p输出指针里面存储的地址值
* 其他指针类型说明，比如float \*p; char \*p;
* 不能乱用类型，比如int a = 10; float \*p = &a;

## 清空指针

* p = 0;
* p = NULL;

# 指针实例

1. void swap(char \*a, char \*b) （注意temp=a; a = b; b = temp;）
2. int sumAndMinus(int a, int b, int \*minus)

# 指针探究

1. 指针变量所占用的存储空间
2. 为何指针变量要分类型？

int i = 2;

char c = 1;

int \*p = &c;

printf(“%d”, \*p);

# 指针与数组

## 指向一维数组元素的指针

## 用指针遍历一维数组元素

* 先遍历char数组(‘i’,’t’)，再遍历int类型数组
* \*(p+i)和\*(p++)的区别
* a+i和a++
* p[0]、p[1]

## 习题

1. 设计一个函数：int arraySum(int a[], int n)，求一维数组a前n个数的和。现在利用int \*p替换int a[]

# 指针与字符串

## 字符串回顾

char s[] = “mj”;

## 其他定义字符串的方式

char \*s = “mj”;

或者

char \*s;

s = “mj”;

## 两种定义方式的区别

* 内存分析
* 画图分析
* 常量和变量字符串的区别
* 常量的内存地址查看

## 习题

1. 编写一个int string\_len(char \*s)，返回字符串s的字符长度

# 返回指针的函数

* 指针也是C语言中的一种数据类型，因此一个函数的返回值肯定可以是指针类型的
* 返回指针的函数的一般形式为：**类型名 \* 函数名(参数列表)**

# 指向函数的指针

## 为什么指针可以指向一个函数？

函数作为一段程序，在内存中也要占据部分存储空间，它也有一个起始地址，即函数的入口地址。函数有自己的地址，那就好办了，我们的指针变量就是用来存储地址的。因此，可以利用一个指针指向一个函数。其中，函数名就代表着函数的地址。

## 指向函数的指针的定义

定义的一般形式：**函数的返回值类型 (\*指针变量名)(形参1, 形参2, ...);**

## 使用注意

* 由于这类指针变量存储的是一个函数的入口地址，所以对它们作加减运算(比如p++)是无意义的
* 指向函数的指针变量主要有两个用途：

 调用函数

 将函数作为参数在函数间传递