# 类型

## 基本类型

整形： int (win86\_64)4字节 (win32)4字节

短整型： short (win86\_64)2字节 (win32)2字节

长整型： long (win86\_64)8字节 (win32)4字节

更长的整形： long long (win86\_64)8字节 (win32)8字节

字符型： char (win86\_64)1字节 (win32)1字节

浮点型(单)： float (win86\_64)4字节 (win32)4字节

浮点型(双)： double (win86\_64)8字节 (win32)8字节

无符号类型： unsigned+类型(以上类型)

## 空类型

Void(特殊类型即未定义的类型)

## 指针类型

普通指针

例:

Int test0=10;

Int\* test1=&test0;

在这里就是将test0的地址存入test1中,使用时只需要在test1前加上\*便可使用

(\*在这里代表解引用)指针的定义便是类型加上\*,右赋值为地址即可

指针数组

Int\* p[10]={0};

指针数组就是指针所指向的数组，由于”[]”优先级大于”\*”所以在这里p是数组名称，数组可存放10个地址，而“int\*“便是指针

数组指针

Int a[10]={0,1,2,3,4,5,6,7,8,9};

Int (\*p)[10]=a；

(在数组中只使用数组名便是地址，可以不用”&”符号来获取地址)

在这里int (\*p)[10]定义了一个指向含有10个元素的一维数组的指针

(注意！！”()”的优先级比”[]”高)

函数指针

Void(\*name)( parameter)

如果在程序中定义了一个函数，那么在编译时系统就会为这个函数代码分配一段存储空间，这段存储空间的首地址称为这个函数的地址。而且函数名表示的就是这个地址。既然是地址我们就可以定义一个指针变量来存放，这个指针变量就叫作函数指针变量，简称函数指针。

## 构造类型

结构体

结构体的声明是在mian函数前进行的定义

*结构体中的数据成员可以是基本数据类型（如 int、float、char 等），也可以是其他结构体类型、指针类型等*

*例:*

*Struct name{*

*Int name0;*

*Char name1;*

*类型 名称;*

*}names;*

*在这定义的names可用作全局变量*

枚举

enum　枚举名{

yellow,

枚举元素1,

枚举元素2,

……};

联合

union name {

int i;

float j;

char str[20];

} names;

在这定义的names可用作全局变量,sizeof(union)的长度为长度为类型中最长的

位域

Struct name{

Int name0：4;

类型 名称：值;

}names;

位域使用是在struct下使用，在类型加上”:值”,值及字节

类

Class name{

Public:

Int name0;Void name3(public){};

};

# 判断

## If语句

If(条件语句){

内容;

}

Else if(条件语句){

内容;

}

Else{

内容;

}

## Switch语句

Switch(条件语句){

Case 结果一:

内容;

Break;//直接退出语句

**continue**;//跳过本次语句

default :

内容;}

# 循环

## For循环

For(int name;条件语句;name++或++name){内容}

## While循环

While(bool值){ 内容}

Bool值即true或false,值大于等于1时为true,可执行内容,反之小于1时为false,跳过内容

# 数组

例：

Int name[10]={0,1,2,3,4,5,6,7,,8,9};

这是个数组,也称为一维数组,输出结果为: 0,1,2,3,4,5,6,7,8,9

Int name[2][3]={{1,2,3}{1,2,3}}

这是一个二维数组,输出结果为: 1,2,3

1,2,3

数组结构: 类型 名称[长度]=赋值

# 函数

## 函数

函数结构:

类型 名称(类型 自定义参数名){

内容;}

例:

void test(int a){

Printf(“%d”,a);

}

Int main(){

Test(1);

}

最后的输出结果就是1，即我们可以创建一个函数,在函数中定义我们的需求,然后在主函数中调用

除void类型外,其余类型均需要返回值（return 值）,返回语句可立即结束此函数,

一般写于函数体最后或错误时返回

## 递归

函数本身是可以调用自己的直到返回出我们想要的结果

int test(int a) {

int s = 0;

if (a==1) {

return 1;

}

else {

s = a + test(a - 1);

}

return s;

}

int main() {

int s=test(10);

printf("%d", s);

return 0;

}

在递归过程中必须要有终点,否则将一种递归,导致错误