# C语言

# 一、引入文件

.c源文件。std标准库，io输入输出库

#include <stdio.h> //导入标准输入输出库

#include "good.h" //调用good.h,直接调用即可

#include "..\two\some.h" //调用上一目录的其他文件夹下的头文件

# 二、样例

// 主函数 一个程序有且只有一个主函数

// 函数的返回值类型 int

//

// argc函数输入的参数个数，

// argv函数输入的参数是什么（文本）

int main(int argc, char \*argv[])

int main(void)

{

printf(“hallo world”);

return 0;

}

//关闭警告

#define \_CRT\_SECURE\_NO\_WARNINGS

# 三、数据类型

常量定义方式

const int PI = 3.14

#define PI 3.14

注意：

1. 通过#define定义的常量是根据值来匹配数据类型
2. const 定义的常量是不安全的，可以通过指针来修改（C语言中）

十进制转二进制(除2反序取余法）

1987 % 2

二进制转十进制（权值法）

第一位：1\*2^0 = 1

第二位：1\*2^1 = 2

第三位：0\*2^2 = 0

第四位：0\*2^3 = 0

... 最后将结果加起来

八进制/十六进制转二进制使用8241法则

八进制转二进制

一位八进制位转换为3位二进制

十六进制转二进制

一位十六进制转换为4位二进制

# 四、编译过程

***C语言代码编译分为4步***

预处理：gcc -E a.c -o a.i

预处理：宏定义，头文件展开，条件编译等，删除注释，不检查语法 （.i)

编译：gcc -S a.i -o a.s

编译：检查语法，将预处理后的文件编译成汇编语言(.s)

汇编：gcc -c a.s -o a.o

汇编：生成二进制文件(.o)

链接：gcc a.o -o a.exe

链接：将需要使用的库链接到可执行文件中

linux平台下更新链接库lld heal.exe

gcc -o file 指定生成输出的文件名

-E 只进行预处理

-S 只进行预处理和编译

-c 只进行预处理，编译和汇编

# 五、system函数

在程序中调用一个外部程序

#include <stdlib.h>

int system(const char \* command);

参数：外部可执行程序名称

返回值：失败 -1

#include <stdio.h>

#include <stdlib.h>

int main()

{

system(“ls -all”);

}

# 六、工具

Qt Creator

windows隐藏cmd一闪

#include <windows.h>

WinExec(“calc”, SW\_HIDE);

VStdio

新建MFC应用-》基于对话框

# 八、计算机中存储的方式

正数反码，补码 源码相同都相同

计算机中数值以补码存储

## 源码，

计算机存储的是二进制数，最高位用于表示数的正负

+0 0000 0000

-0 1000 0000

## 反码：

负数取反（除第一位）

## 补码（用于计算加法，正数加负数）

反码最后一位+1

会出现溢出，系统自动舍去溢出数据

sizeof获取的结果是字节（Byte)

# 宏定义

#ifndef \_\_luck\_\_MainMenu\_\_

#define \_\_luck\_\_MainMenu\_\_

// luck项目名称，MainMenu文件名称

#endif

# 3、BASE

## 变量

int 整形 %d

float/double 浮点型 %f

short 65535-长度

long

字符类型：char

每一个字符都有自己的编码ACSII，如果字符串按数字输出，将输出ACSII编码

char a = ‘12’ %c

scanf（“%d”）获取用户输入数据

## 强制转换

int a =123;

float b = (float)a

## 三目运算符

条件？真，输出：假输出

# 指针与数组的关系

数组名是数组的首地址，是一个常量，也可看成一个指针

int arr[] = {1,2}

int \*p = arr

printf(“%p”, arr) //地址

int \*p = arr;

for(int i=0, i<3, i++)

{

printf(“%d”, arr[i])

printf(“%d”, p[i])

printf(“%d”, \*(p+i))

}

# 一维数组(连续空间）

数组本身就是指针

int Array[20] //定义20长度的数组

//获取数组的个数

//先知道int/char等变量的大小sizeof(int)

int len = sizeof(Array)/ 4

//初始化的时候赋值

int Arr[10]={0,1,....} //定义一个10个大小的数组,连续空间

int Array[]={1,2,3,4} //定义并赋值

//直接

printf(“%d”,Arr[1]) //输出

# 一维数据内存存储

根据操作系统与定义的数据类型不同，数据占用的空间不同，比如int是4位（byte)

数组是内存中连续存在的空间，（VS中调试显示的是16进制，并有数据对齐等问题）

printf(“%p”, &Arr[1]) //以无符号十六进制格式打印

直接打印数组名，会打印数组的首地址

数组名=数组的第一个元素，下标0

# 二维数组

# 字符串数组

%s会接受\0之前的所有字符

Char name[]=”holle” //字符串，有\0

Char name[]={“h”,”o”,”l”,”l”,”e”} //字符串数组，**没有\0**

// \0字符串结束标准，字符串的内容要循环修改，将字符一个一个赋值，不能直接等于赋值

char name[20]={‘a’,’b’,’\0’}

char \* name = “字符串常量”

# 产生随机数

#include<time.h>

#include<stdlib.h>

int main()

{

#添加随机数种子,不添加随机数种子，随机的数都是一样的

srand((unsigned int)time(NULL))

rand() //产生随机数

rand() % 100 //0~99的随机数

rand() % 11 + 50 //50~60之间的数

rand() % 10 + 50 //50~69之间的数

}

# 使用内置函数拷贝

//主要是因为，C语言中字符串是存放在数组中的，拷贝字符串需要把数组中每一个元素拷贝一遍

#include <string.h>

strcpy(a,b)

# 字符串处理

#include<stdio.h>

对于未知大小数据，不能超过字符数组大小，不然缓存溢出

char ch[100]; //数组必须有大小

char \* gets(char \* ch)

从输入设备上获取y一个字符串，直到换行或\0结束

gets(ch) ; //将输入的字符串放到ch中,会接受任意大小字符，不安全

char \* p = gets(ch); //p与ch的结果是一样的

gets与scanf的区别：gets可以输入空格

puts()

输出一个字符串并换行

puts(ch);//带换行符

int vlu = puts(ch) //vlu不为-1都为成功

char \* fgets(char \*s, int size, FILE \* stream)

始终会将\0加入

文本操作stdin标准输入库,size接收的大小

fgets(ch, 100, stdin)//将输入字符放到ch字符指针上

当输入的数据大于fgets的ch字符串指针能接收数据时，fgets将把多余的数据都扔掉，留下一个内存接收\0

int fputs(const char \*str, FILE \* stream)

stream固定为stdout

输出\0前的所有字符

## scanf格式化

scanf(“%[^\n]”, arr)

scanf(“%d %d”, &a, &b)

scanf(“%1d”, &a) //接收1位数据

scanf(“%1s%2s”, arr, arr2) //接收指定字符

scanf(“%\*s%d”, &a) //\*忽略字符

scanf(“%\*[123456]%c”, &a) // 忽略123456任意字符

scanf(“%\*[a-z]%c”, &a)

printf(“%.2f”, 3.145) //保留2位小数

printf(“%5d”, 12) //一共显示5位数

printf(“%4s”, “ab”) //显示4位数字

printf(“%.4s”, “ab”) //截取

## strlen

读取到\0结束

include<string.h>

char arr[] = “hello”

int len = strlen(arr) //字符串有效长度，忽略\0

与szieof区别是字符数组的长度

## char \* strcp(char \*dest, const char \*src)

从src复制到dest

目标字符串，源字符串，字符串长度,有限复制，不会复制\0

strncp(dest, src, 5)

## char \*strcat(char \* dest, const char \*src)

将src字符串追加到dest变量中

char arr1[] = “hello”;

char arr2[] = “word”;

strcat(arr1, arr2) //这里数组下标越界，需要修改arr1的数组大小

strncat()有限追加

strncat(arr1, arr2, 3)

有限追加会将\0加入

## strcmp(const char \*s1, const char \*s2)

比较字符串，比较的是ASCII吗，相等会返回0，不同的操作系统，返回值不同，比较\0之前的所有有效字符

strncmp(arr1, arr2, 2)//比较几个数之前的数

## strchr(const char \*ch, int c)

查找字符第一次出现的位置

char \*p = trrchr(arr, “l”); //p的结果为从l到\0的字符

失败返回NULL

strstr()//在一个字符串中查找另一个字符串

## strtok()

会修改原始字符串

char a[] = “www.baidu.com”;

char \*p = strtok(a, “\*”)

p // www

while(p!=Null)

{

p = strtok(NULL, “.”); //

}

## atoi()

将数转换为数字

atof()

atol()

字符串分割

# 函数

1、有参数无返函数

声明

extern int add(int a, int b);

//定义

int add(int a, int b)

{

return 0; 必须有返回值

}

//调用

int main()

{

add(1,2);

}

# 多文件编程

1. 在同一个文件夹中，所有函数都是全局函数，所以，在任意文件中定义一个函数，均可以被其他方法调用,
2. 使用.h文件，.h文件用于声明函数所在的位置

// 函数、变量的声明

// 系统库的调用

extern int max(int a, int b);

1. 主函数引用.h文件

编辑器会自动引入文件，对于非编辑器使用命令

gcc -o 可执行程序.exe main.c 文件.c 头文件.h

## 防止头文件被多次包含

#pragma once //版本高

# SOMEFILE可定义

#ifndef \_\_SOMEFILE\_H\_\_

#define \_\_SOMEFILE\_H\_\_

// 声明语句

#endif

# 指针

## 说明

无论是什么类型得指针，村的都是地址，内存地址都是无符号整形，都是4字节大小

\* 取值

& 取地址

int a=5,b,c

int \*p //int类型的指针\*p，4个字节

p = &a //p中放的是地址

\*p = 15 // 将p指针指向得a变量重新赋值为15,相当于a=15

c = \*p //将p中的值赋值给c

b = &a // 将地址赋给b,但b不是指针变量，不能使用\*

void \* p = &a //万能指针，void

\*（int \*)p = 100 //当取值得时候要进行类型转换

**指针变量存放非地址数据没有实际上的意义**

p=a // p保存的是a中的值，相当于普通变量

printf(\*p) //这里会溢出，因为P变量存的值是5,\*p是找 内存中指向5的位置

**指针赋值**

\*p=a / p=&a //这时2种方式，都是获取a的地址，变量p中保存的都是地址

printf（\*p）

**溢出**

\*p=&a // 这里会溢出，不知道指到哪里了

**\*必须是指针类型才能使用**

int p, b=5;

p=&b; //a中存放地址

printf(\*p); //语法错误，只有指针类型才能使用 \*

printf(p); // 打印b的地址

**指向数组的指针-数组本身就是指针**

int stu[2] = {1,2}, \*p

// 这3中写法都指向了数组stu，stu存放的是数组的首地址，

// stu[0]存放的是首元素，so，要使用 &stu[0]

p = &stu[0] / p=stu / \*p=stu

printf(\*(p+3)) // 指针下滑，指向第3个数

\*((int \*)p +1) =200 //这里因为先转换为了int类型，所以应+1即可

**情况八-指向函数的指针**

int func(int a){}

int (\* pfun)(int a)； // 定义一个函数指针

pfun=func; // 指向一个函数

pfun=&func; // 指向一个函数

printf（pfun（1）） // 调用函数

指针和指针参数

void tab(int \*a, int \*b) //地址传递，改变实参的值

{

int tmp = \*a;

\*a = \*b;

\*b = tmp

}

int main()

{

tab(&a,&b) //这里会修改a,b的位置，作用域不光是tab函数中

}

**字符串常量**

不能修改，与程序有相同的运行周期

char \* arr = “Homeworld” //这个变量不允许修改

char a[] = “hallow”

char \* p = a;

p[0] = “x” // xallow

\*p = “x” // xallow

p++

\*p = “x” // xxllow

printf(“\*p=%s”, \*p) //\*p=xxllow

printf(“p=%X”, p) // p=0xff01

# 内存管理

**作用域**

int a ; //作用域为{}之间，变量存放在栈中，栈的顺序是从上到下存储，声明不会分配内存空间，定义的时候才会分配内存空间

生命周期

局部变量 int a = 10 函数结束

全局变量 int a = 10 程序结束

静态变量 static int a = 10 程序结束

代码区：

程序指令

数据区：

初始化数据

1. 初始化的全局变量
2. 初始化的静态局部变量
3. 初始化的静态全局变量

未初始化的数据

1. 未初始化的静态局部变量，默认值为0
2. 未初始化的全局变量，默认值为0
3. 未初始化的静态全局变量，默认值0

字符串常量、

栈区

变量、数组（只有C语言放到栈区）、结构体、指针

栈区大小在不同的操作系统中，系统分配给每一个栈区大小空间不同，windows 1M-8M不等

linux 1M-16M

堆区

音频文件，文本文件，视频文件

如果创建大的数据，应该放到堆区

死循环只能沾满CPU，不能沾满内存

## 开辟内存空间（堆空间）

//开辟指定大小内存

int \* p = (int \*)malloc(1000 \* sizeof(int))

**内存布局**

# const int

使用const定义的值是不安全的,因为可以通过指针修改

#define LV 100 //这样定义是无法修改

const int \* p //不能改变指针指向的数据，但可以修改p中的值（地址）

int \* const p = &a //可以修改指针指向的数据，但不能修改p的值（地址）

## 6、结构体

结构体的存在是为了节省代码，相同的变量，不用多次声明，通过结构体声明多个具有相同属性的变量

**结构体变量，结构体数组，结构体指针**

### 定义方式一

1. **声明结构体**

struct Player{

int a;

}

1. **定义结构体 p1**

struct Player p1; // p1包含Player中声明的所有变量

**3、使用结构体变量赋值**

p1.a = 1; // 给结构体的属性赋值

### 定义方式二

struct Player{

int a;

}p1,p2 // 定义了2个结构体变量（Player）:p1,p2

struct Player p3;

struct Player p4;

-----------------------------------------------------------------

// 可以这样定义，但这样定义将不能定义p1,p2之外的结构体变量

struct { //这个结构体并没有标明其标签

int a;

}p1,p2

### 定义方式三

typedef struct { // 用typedef创建新类型,新类型名为Player

int a;

}Player

Player p1; //定义一个结构变量

p1.a = 1;

---------------------------------------------------------

typedef struct \_Player{

int a;

}Player;

Player p1;

p1.a = 1; // 给结构体的属性赋值

struct \_Player p2 //结构体定义

### 指针使用结构体

typedef struct \_Player{

int a;

}Player;

Player \* nowp //声明一个**机构体**指针，这个指针（nowp）指向结构体（Player）

nowp=&p1 // 将结构体指针，指向p1

nowp->a // -> 指针的成员操作符与.意思相同

### 结构体数组-没确定使用方式

typedef struct \_player{

int a;

}Player;

Struct Player info[30] // 使用结构体Player 声明一个30容量的info

info[1].a=1;

## 7、联合体，结构体数组

typedef struct \_Play{

int a;

union { // 联合体，类似枚举类型,只保存联合体中最大的容量

int ID;

char name[20];

}job;

}

## 8、枚举类型

enum SYS{ // 枚举类型的值是根据顺序排列的

ADD=1, // 1

QUERY=2, // 2

}

enum SYS input; // 声明一个枚举变量input

scanf(“%d”, &input) //输入信息放到input中,提示用户输入1、2、3、4

if (input == ADD) {} // 枚举类型会自动管理相关变量

## 9、动态分配空间

# C语言-Player结构体动态分配空间

Player \* None=（Player \*）malloc(sizeof(Player));

#

Player \* None = new Player;

free //释放空间

## 10、文件操作

FILE \* f1; //定义一个文件指针

f1=fopen(“路径”, “w”)

## 11、杂项

变量存储内省

堆区 存放动态分配的数据

栈区 存放局部数据，局部变量

全局数据区 全局数据和静态数据，全局变量

程序代码区 程序的各个函数代码

变量存储类型

Auto自动类

register寄存器类 仅局部变量

static静态类

extern外部类

# 二、C++

## 1、hello world

#include <iostream>

int main(int argc, const char \* argv[]){

// 标准的命名空间std

std::cout << “hello world”;

}

## 2、标准库

对象使用.

std::vector<>;

std::cout;

std::endl;

## 3、类

### （1）定义类

**NPC.h文件中定义函数**

使用宏定义防止多次加载

#ifndef \_\_xxx

#define \_\_xxx

class Rect{}

定义类

class NPC

{

NPC（）； //构造方法,与类名相同

~NPC（）； //析构方法

Rect \* a; //引用另外的类

private:

protected:

public:

string Name；

void move(int a); //定义一个方法

}

#endif //宏定义

### （2）实现类

**NPC.cpp文件实现方法**

#include “NPC.h”

NPC::NPC(){} //实现构造方法

NPC::~NPC(){ //析构方法

delete this->a; //释放空间a

}

void NPC::move(int a){ //::成员操作符

std::cout << “1”;

}

### （3）调用

**main.cpp**

#include “NPC.h”

int main()

{

NPC n1 // 创建一个类，分配空间

n1.move(1) //调用类中的方法

#-------- 动态生成一个对象

NPC \* n2；

n2=&n1 // 可以指向一个已经创建的对象

n2=new NPC() // new一个对象

#-------------

NPC n3(1) //调用构造函数

NPC \* n3 = new NPC(1)

}

1. 构造方法
2. 析构方法

## 4、方法重载

在同一个类中有相同的名称，但参数不同(包含类型不同)，系统会根据参数不同，调用不同的方法

class A

{

void B(int a){};

void B(int a, int b){}; //重载

}

## 5、类的继承

class NPC{}

#include “NPC.h”

class Other:NPC{} //继承NPC

class Other:NPC,NPC2 {} //继承多个

class Other:public NPC {} //公有继承

多层继承会依次调用各个层的构造方法

## 6、虚函数-抽象类-不懂

虚函数永远执行子类中的方法（方法重载）

实现父类指针调用子类方法

class NPC2:NPC

{

virtual void fire(int a); //虚函数定义

virtual void fire(int a)=0; //纯虚函数，子类必须定义

}

class Sound{

virtual void play()=0;

}

void sound::play(){}

#include “sound.h”

## 7、操作符的重载

class NPC

{

void operator++(); //重载++操作符

}

void NPC::operator++(){} //重新实现++操作符

## 8、模板-泛型编程

### 模板函数

//定义一个模板函数

// 函数名getMax

template<class T>

T getMax(T a, T b) {

return a>b?a:b;

}

int main()

{

// 不用特殊定义属性，模板函数会自动返回T类型的值，输入T类型的变量（T类型，是根据你的输入自动变换的）

std::cout<<getMax(1,2)<<std::endl;

std::cout<<getMax(“a”,”b”)<<std::endl;

}

### 模板类-不懂

模板定义的变量是可变的，其他的都一样

template<class T>

class LinkList{

struct node{

T \* data;

}

}

## 9、C++11 新特性

### 1、auto 自动实现类型转换，

auto 可以指向任何函数，实现类型转换【弱类型语言】

auto 定义的就是，任意类型语言，可以放任何东西

### 2、增强的for循环

**[可以理解成for...in]**

for(auto tmp:allNPC){

//tmp是allNPC中的每一个值

tmp->move(30,40) //移动

}

### lambda 匿名函数

[] 使用外部变量

auto a = **[**=赋值方式、&引用方式**](**参数**){**代码块**}**

a(); //调用

# C++内存管理对象

new 创建对象、delete删除对象

# 字符串

**字符串都是数组，char类型也是数组（这么理解）**

字符串的二维数组：C语言与其他语言不同

**普通定义-字符串常量中放的是变量**

char a[] = {“221”};

char a[2][10] 表示有2个字符串，每一个字符串的长度是10.

**指针定义-a中放的是地址**

char \*a[] = {“11”,”22”};

用处：

当在使用if的时候，