C 语言学习笔记

--zhangsj

C 语言发展史：

1960 原型 A 语言 -> ALGOL 语言
1963 CPL 语言
1967 BCPL 语言
1970 B 语言
1973 C 语言

C 语言特点：

1、基础性语言
2、语法简洁，紧凑，方便，灵活（指针的作用）
3、运算符丰富，数据结构丰富
4、结构化、模块化编程思想
5、移植性好，执行效率高
6、允许直接对硬件操作

C 语言学习建议：

1、概念的正确性
2、动手能力
3、阅读优秀的程序段
4、大量练习，面试题

C 课程讲解思路：

1、基本概念
2、数据类型，运算符和表达式
3、输入输出专题
4、流程控制
5、数组*
6、指针*
7、函数*
8、构造类型
9、动态内存的管理
10、调试工具和调试技巧（gdb,make）
11、常用库函数

平台介绍：win10 + 64 位 Ubuntu 20.04 wsl2, vscode + Wsl-Remote, gcc(make)

hello.c:

编译器：gcc
C 源文件-预处理-编译-汇编-链接-可执行文件
预处理：gcc -E hello.c > hello.i (重定向保存，预处理文件后

缀为.i)

编译：gcc -S hello.i (默认产生后缀为.s 的目标文件)
汇编：gcc -c hello.s (默认产生后缀为.o 的目标文件)
链接：gcc hello.o -o hello (指定生成可执行文件的名字)
可执行文件：./hello 执行当前目录下的可执行文件 hello

省略用法：

gcc hello.c (默认生成可执行文件 a.out)
gcc hello.c -o myhello (指定生成可执行文件的名字)

make 用法：

make hello (生成可执行文件 hello)

一、基本概念

1、以 helloworld 为例对写程序的思路提出以下要求：

1）头文件包含的重要性 (gcc hello.c -Wall 调试方法)
2）以函数为单位进行程序编写
3）声明部分 + 实现部分
4）return 0;
5）多用空格和空行
6）适当添加注释

2、算法：解决问题的方法（流程图，NS 图，有限状态机 FSM）
3、程序：用某种语言实现算法
4、进程：32 位环境下一个进程最多占用 4G 空间
5、防止写越界，防止内存泄漏，谁打开谁关闭，谁申请谁释放

二、数据类型，运算符和表达式

1、数据类型：

1）基本数据类型（short int long float double char）
2）构造类型（array struct union enum）
3）指针类型
4）空类型（void）

作以下思考：

1）不同数据类型所占字节数（标准 C 并未严格规定各类型所占字节数）

2）存储区别（signed unsigned）
3）不同类型的数据间进行转换（隐式，显式转换）
4）特殊性：
（1）布尔型 bool
（2）float 类型数无法和一确切的数比较是否相等（float 类型本身并不精确）
（3）char 型是否有符号并未定义
（4）不同形式的 0 值 (0 '0' "0" '\0' NULL)
（5）数据类型与后续代码中所使用的输入输出要相匹配

2、变量和常量：
常量：在程序执行过程中值不会发生变化的量
分类：整型常量，实型常量，字符常量，字符串常量，标识常量
1）整型常量：1790，34，56
2）实型常量：3.14，2.56，0.67
3）字符常量：单引号引起来的单个字符或转义字符，
如 'a','D','\n','\0','\ddd'(三 位 八 进 制数),'\xhh'(两位 16 进制数)
'\015','\x7f','\018'(非法!)
4)字符串常量:双引号引起来的一个或多个字符组成的序列，
如"helloworld","a",""(空字符串，只有一个'\0'占用一个字节)
"abcd\n\021\018"(特殊)
5）标识常量：#define，处理在程序预处理阶段，占编译时间，
优点是一改全改，缺点是不检查语法，只是单纯的宏体与宏名之间的替换

变量：用来保存一些特定内容，并且在程序执行过程中值随时会发生变化的量
定义:【存储类型】 数据类型 标识符 = 值
TYPE NAME = VALUE;
标识符：由字母，数字，下划线组成且不能以数字开头的一个标识序列，取标识符尽量做到见名生义
数据类型：基本数据类型 + 构造类型
值：注意匹配

存储类型:auto static register extern(非定义型关键字，属于说明型关键字)
auto：默认，自动分配空间，自动回收空间
static：静态型，自动初始化为 0 值或空值，并且其变量的值具有继承性

常用于修饰变量或函数

register：寄存器类型(建议性关键字)

register int i = 1; 由编译器决定是否存储在寄存器中，

大小有限制只能用来定义局部变量，32 位机器只能定义 32 位大小的数据类型，如 double 就不可以，

寄存器没有地址，所以无法打印寄存器类型变量的地址进行查看或使用

extern：说明型，不能改变被说明的变量的值或类型

变量的生命周期和作用范围：

全局变量：作用范围从定义位置开始直到程序结束

局部变量：作用范围从申明位置开始直到当前块作用域结束

1）全局变量和局部变量

2）局部变量和局部变量

3、运算符和表达式

表达式和语句的区别：

i = 1 表达式

i = 1; 语句

运算符部分：

1）每个运算符所需要的参与运算的操作数个数

2）结合性（单目运算符，条件运算符以及赋值运算符这 3 种运算符右结合）

3）优先级

4）运算符的特殊用法

如：%要求左右操作数必须为整型， == 和 = 注意区别，逻辑运算符(&&, ||)短路特性

5）位运算的重要意义

将操作数中第 n 位置 1，其他位不变：num = num | 1 << n;

将操作数中第 n 位清 0，其他位不变：num = num & ~(1 << n);

测试第 n 位：if(num & 1 << n)

从一个指定宽度为 w 的数中取出第(m -> n)位：(num << (w-n)) >> (w-n+m)

三、输入输出专题

input & output -> I/O (标准 IO，文件 IO)

1、格式化输入输出函数：scanf,printf

int printf(const char *format, ...);

format："%【修饰符】格式字符"

判断是否变参实现的方法：多传参数如果报语法错误则为定参，如果不报而在使用过程出问题则为变参

int scanf(const char *format, ...);

%s 作为输入项非常危险：(见 scanf.c 文件)

%s 作为输入项时中间不能添加分隔符(tab，space，enter)

%s 不会提示越界问题

scanf 放在 while()中非常危险，要注意能否接收到正常有效的内容，校验返回值！

format：抑制符*，如：scanf("%*c%c", &ch);

2、字符输入输出函数：getchar,putchar

int getchar(void);

int putchar(int c);

3、字符串输入输出函数：gets(!),puts

char *gets(char *s);

gets 函数十分危险，可以用 fgets，getline 替代

int puts(const char *s);

四、流程控制

顺序，选择，循环

NS 图，流程图

简单结构和复杂结构：自然流程

顺序：语句逐句执行

选择：出现一种以上情况

循环：在某个条件成立情况下重复执行某个动作

关键字：

选择：if-else switch-case

循环：while do-while for if-goto

辅助控制：continue break

1、顺序：

if-else：

格式：if(exp)
  cmd;
或：
if(exp)
  cmd1;
else
  cmd2;
注意：else 只与和它最近的 if 相匹配

switch-case：
格式：switch(exp)
{
  case 常量或常量表达式:exp1;
    break;
  case 常量或常量表达式:exp2;
    break;
  ...
  default: exp;
}

2、循环：while do-while for if-goto

while:
格式：while(exp)
  loop;
最少执行 0 次

do-while:
格式：do
{
  loop;
}while(exp);
最少执行 1 次

for：
格式：for(exp1; exp2; exp3)
  loop;
最少执行 0 次

if-goto：
(慎用：goto 实现的是无条件的跳转且不能跨函数跳转)

3、死循环：

while(1);
for(;;);

杀掉死循环：ctrl + c

4、辅助控制关键字：break, continue

五、数组

构造类型之一，连续存放

一维数组

1、定义

【指定存储类型】 数据类型 标识符[下标(整型常量或整型常量表达式)]

2、初始化

不初始化

全部初始化

部分初始化

static

3、元素引用

数组名【下标】

4、数组名

数组名是表示地址的常量，也是数组的起始位置

5、数组越界

二维数组

1、定义，初始化

【指定存储类型】 数据类型 标识符[行下标][列下标]

2、元素引用

数组名[行标][列标]

3、存储形式

顺序存储，按行存储

4、深入理解二维数组

字符数组

1、定义，初始化，存储特点

【指定存储类型】 数据类型 标识符[下标]

初始化：

单个字符初始化

字符串常量初始化

2、输入输出

gets(), puts()
scanf(), printf(): %s

3、常用函数

strlen & sizeof
strcpy & strncpy
strcat & strncat
strcmp & strncmp

多维数组

六、指针

1、变量与地址
2、指针与指针变量

指针就是地址常量

3、直接访问与间接访问
4、空指针与野指针
5、空类型

```
void* p = NULL;
```

6、定义与初始化的书写规则
7、指针运算

&p, *p, 关系运算, p++, p--

8、指针与数组

指针与一维数组
指针与二维数组
指针与字符数组

```
1) char *str = "hello";  str = "world";  puts(str);
2) char str[] = "hello";  strcpy(str, "world");
puts(str);
```

9、const 与指针

指针常量：

```
int *const p;
```

常量指针：

```
const int *p;
int const *p;
```

```
const int *const p;
```

10、指针数组与数组指针

数组指针:【存储类型】 数据类型 (*指针名)[下标] = 值

如：int (*p)[3]; --> type name; --> int[3] *p;

指针数组:【存储类型】 数据类型 * 数组名[下标] = 值

如：int* arr[3]; --> type name; --> int*[3] arr;

11、多级指针

七、函数

1、函数的定义

数据类型 函数名 (【数据类型 形参名，数据类型 形参名，...】)

2、函数的传参

值传递

地址传递

全局变量

3、函数的调用

嵌套调用

递归

4、函数与数组

5、函数与指针

指针函数

返回值 *函数名(形参表)

如： int *func(int)

函数指针

类型 (*指针名)(形参表)

如： int (*p)(int)

函数指针数组

类型 (*数组名[下标])(形参表)

如： int (*arr[N])(int)

指针函数指针数组

类型 *(*数组名[下标])(形参表)

如： int *(*p[N])(int)

八、构造类型

1、结构体

1)产生原因及意义

2)类型描述

```
struct 结构体名
{
    数据类型 成员 1;
    数据类型 成员 2;
    ...
};        //结构体定义就好比 int，其不占存储空间，仅是类型描述
```

3)嵌套定义

4)定义变量(常规变量， 数组， 指针)，初始化及成员引用

成员引用：变量名.成员名

指针->成员名

*(指针).成员名

5)结构体内存占用空间大小

6)函数传参(值， 地址)

2、共用体(可以提供多个属性，但同一时刻只能存在 1 个属性生效，类比性别)

1)产生及意义

2)类型描述

```
union 共用体名
{
    数据类型  成员名 1;
    数据类型  成员名 2;
    ...
};
```

3)嵌套定义

4)定义变量(变量，数组，指针)，初始化及成员引用

成员引用：变量名.成员名

指针->成员名

*(指针).成员名

5)占用内存大小

6)函数传参(值，地址)

7)位域

union

```
{
    struct
    {
        char a:1;
        char b:2;
        char c:1;
    }x;
    char y;
}w;
```

3、枚举

```
enum 标识符
{
    成员 1,
    成员 2,
    ...
};
```

九、动态内存管理

malloc calloc realloc free

void*和函数指针进行转换在 c99 标准中未定义

原则：谁申请谁释放

typedef: 为已有数据类型改名

typedef 已有数据类型 新名字;