Simi Documentation / C

Author: Craig Brown
Version: 1.0.0
Date: Feb 2, 2025
Source: https://github.com/saidake/simi-docs/tree/master/pdf

C / Concept

MINGW

预处理　将头文件写入包含的文件中　(test.i　)
编译　生成汇编代码　(test.s　)
汇编　生成二进制代码　(test.o　)
链接　链接库文件生成可执行文件　(test.exe)

进制

十进制 Decimal（D）　二进制 Binary（B）　八进制 Octal（O）　十六进制　Hexadecimal（H）　【无论几进制计算，溢出都向前进 1】　【二进制：1001　可以表示 2^4=16 种状态，16 个数】

进制算法　8 进制表示　0218　= 2 * 8^2　+　1* 8^1　+　8* 8^1

16 进制加法　0x58+0x76=0xce　（满 x 进位，余数留在原位置）

原码，反码，补码　1000 0111　1111 1000　1111 1001　（取反<尾 1>不变，排除符号位，补码的补码是原码）　【有符号数的负数符号位 1　不会发生变化，而正数的 0 会】

1101 1101　1010 0010　1010 0011　（在计算机存储中，正数用原码表示，负数用补码表示）

1001 1111　1001 0000　1001 0001

1100 0000　1011 1111　1100 0000

1000 0000　1111 1111　1000 0000

2^{31}=　2147 483648

2^{32}=　4294 967296

标识符

由字母 下划线 数字组成　不能数字开头

程序开始

```
#include<stdio.h>
int main(){
    return 0;
}
```

C / Core

Variables

```
int a=999, b=a/(a+=100)
int a=0b11001
int a=-0b11001
int a=-017
int a=-0x18
int a=2e-5
```

整数（4byte）【进制表示时没有符号位，需要开头加 -，转换成十进制计算输出】

```
short a=017, b=a/(a+=100)
short a=0x19
```

短整数（2byte）

```
long c=44LU, b=c/(c+=100)
long a=0x98lu
```

长整数（4byte）【后缀 l 或 L，可搭配 u 或 U，后缀不匹配输出 0】

```
float a=0.66f
float a=4.4E-10
```

单精度浮点数（4byte　1bit 符号位 8bits 指数位 23bits 尾数位）　【后缀 f 或 F】

```
double a=2.988L
```

双精度浮点数（8byte）　【后缀 l 或 L】

```
char a=97
char a='z'
const char*= "rre"
char a[]="rre"                                  字符（1byte）  【可用为数字，ASCII 码存储：  48（0）  65（A）  97（a）】
unsigned int a=17                               无符号数 【范围扩大，-2147483647 ~ -1  0 ~ 2147483647  ---> 0 - 4 294 967 295】

string a="xxx"
string a="xxx"
bool a=true
```

int a[]={1, 2, 3}; char a[3]="fa" 行数可省，却值默认为'\0' 【数组名为首元素地址，未初始化赋值为随机数，字符串数组无法修改内部值但可用 strcpy，不能通过 a++等运算符操作 数组地址】

int a[][3]={{2，3，4}，{4，6，7}}；char a[2][3]={{"fa"},{"do"}}; 行数可省，却值默认为'\0' 【定义时[]里不能放变量，只能包含一种数据类型，a[2]==&a[2][0]代表此行一维数组地址】

```
#define LD1 {clrAqua, \          Escape sequences
             clrGray, \
             clrGray}
```

TIPS：从左侧编译，从右侧赋值 // int a=b=c=0; this.d=this.e=2 错误，bc 没有声明
整数前缀：二进制 0b，八进制 0，十六进制 0x
整数后缀：无符号 U

Constant		
const int xx=3;	限制初始化赋值一次	
const int* xx=&a;	多次赋值	无法修改 指向变量 a
int* const xx=&a;	限制初始化赋值一次	可修改 指向变量 a
const int* const xx=&a;	限制初始化赋值一次	无法修改 指向变量 a

Data Type

char（1byte 可用为数字） ASCII 码存储： 48（0） 65（A） 97（a）
"reer" （字符数+1 byte 末尾默认有一个空字符'\0'，整个字符串代表首字母的地址且地址必须为 const，字符串中可含转义字符）
扩展类型 complex.h 复数类型 complex double a =3.0+4.0*_Complex_I; --> 3+4j
转义字符 \u UNICODE 编码（接 16 进制） \n 换行 \r 回车 \’ 单引号） \” 双引号 \ \反斜线 \?问号 \0 空（显示为空格==0） \t 制表 \ddd（八进制） \xdd（十六进制）
【ASCII 码直接表示字符，不存在十进制书写】
sizeof(int) 返回数据类型或变量的占字节数 （溢出数据范围会旋转取值，0 算入正数范围。 对于 int 类型 4byte： 0 ---> 2^{31}-1 ---> -2^{31} ---> -1 2^{31}+1= -2^{31}）
int(999) 临时变量，表达式结束就被清除
register int n 尽可能将变量存在寄存器中，可能会失败（长度小于等于整型长度，不能&取地址，编译器大多会忽略 register）

TIPS：二进制不足 4byte 的在前面补 0（负数最前方补 1） // -0b101 --> 10000000 00000000 00000000 00000101

Type Conversion

自动转换（向高类型转换）　short，char → int → long → double ← float　（signed → unsigned）　// 1/3.33--->1.00/3.33

强制转换　(int)x　不影响原变量，超出类型的范围旋转取值

Operator

优先级一级　[ ] 数组下标　() 圆括号　. -> 获取成员

优先级二级　-负号　~ 位取反　++ --自增自减　* & 地址取变量 取地址符　!非　(int) 强制转换　sizeof　【同级先运算右侧】【++运算符所在的式子计算完毕后再加 a=1,a++,b=a　b=2】

算术运算符　* / %　>　+-　c=A%B　c 总是小于 B（去掉 A 里 所有的 B　最后剩余的数）【　(xx++)++不能二次延迟，++1 不能修改固定值】

左移右移　<< >>　0b101<<3=　向左移动，溢出位移除，移入位 0　【部分编译器右移有所不同：　逻辑移位（移入位 0）　算术移位（移入原符号位）】

关系运算符　> >= < <= > == !=　结果为 01　（0 为假，非 0 为真）

位运算符　&　>　^　>　|　0b101&0b110=0b100 与　0b100 | 0b110=0b110 或　-0b100^0b110=-0b010 异或　~-0b10=1 位取反（补码取反）

逻辑运算符 &&　>　||　结果为 01（优先级：　算术运算符 > 关系运算符 > 逻辑运算符）【　jj&&123&&DD（真---最右值）　JJ||123||DD（真---最左值）　】

条件运算符　a>b?x:y　true?false?true?true1:2:3:4:5　==4　flase 出现终止判断，有多少 true 从后向前推多少位（ 冒号内容必须比问号内容多一个，内部不能有 return）　【同级先运算右侧】

赋值运算符　= *= /= %= += -= <<= >>= &= ^= |=　（赋值表达式的值：赋值后的最左值）【同级先运算右侧，左侧只能为变量】

逗号　,　（逗号表达式的值：最后一个表达式的值）

表达式：创建和处理变量（一般表达式用于括号内）　// int a=（a, b, c+d）　-->　a=c+d　此逗号表达式内含有算术表达式

位运算：　负进制数都用补码计算，不足 4byte 的在前面补 0（最前方补 1）　// -0b101　-->　10000000 00000000 00000000 00000101

<<左移右移　正数原码计算，负数补码计算　printf 只有%d 能转换　位运算后的补码（%o%x 会直接识别移位或位取反后的补码）　//-0b010<<1　==　-4　37777777774　fffffffc

& 位与位或　正负数都原码计算

TIPS：　随机数范围　Math.random() * (max - min + 1) + min

Statement

条件　if（）{ }　{ }可以看成整合的一条语句，只有一条语句时{}可省

if（）{ }　else if( ){ }　else{ }　三条语句中只执行一条满足条件的语句

switch (typeof A){　case 0: xxxx; break;　default: xxxx; break;　}　匹配一个 case，直到遇到 break 才停止执行，所有 case 都不满足则执行 default

【不能在 case 代码块中定义变量，判断整数 A 是否等于各个 case 整数值】

switch 内部如果用了 return　那么 switch 内部所有的语句将不会执行

循环　while( ){　}　while，for 循环条件默认为真

do{　}while( );

for(　; A ; B){　}　A-->{　}-->B　三个表达式都可省略　【第三个表达式可以为不带分号的长代码 for(;;　a++,b=a/(a+100),cout<<"c")　for(;;); 引号单独为空语句】

break 跳出当前循环（不包括外部循环）　continue 跳过循环到条件括号（for 循环里跳到第三个表达式，continue 只对循环有效，

对 if 无效，）

TIPS: if(fun()) 判断括号内可以调用函数

for(; !hasCommon(md,min,max); md+=max;) for 内有函数时，第三个表达式不能操作与形参名相同的变量

C / Features

作用域

文件作用域 > 大括号内 大括号外

extern int x; （提前激活 本文件或其他 c 文件的变量/函数） 【extern void fa(); 等效于 void fa(); 】

激活变量需严格对应变量声明格式 char a[]="dog" extern char a[];

static int x; 存储在静态区（初值默认为 0） 声明和定义同时进行

全局静态变量只能在当前.c 文件上使用（激活也无效）

Function

definition

void <function-name>(int <variable >, int <variable>){ } Header files are not suitable fro defining functions, function should be declared.

激活函数（提前激活 本文件或其他 c 文件的函数，形参和实参都占用内存空间） 【主函数内外都可以】

变量 int c 【 int int a】

普通地址 int *p 【 int * int *a int a[] int [] int a[999] int [888] 】

指针数组地址 char *a[10] 【 int **a int *a[] int *a[999] 】

二维数组地址 int (*p)[10] 【int (*)[10] int (*p)[10] int a[][10] int [][10] int a[999][10] int [888][10]】

void main(int argc, char **argv) argc（命令行输入参数 个数） argv（命令行输入参数 字符串） 从控制台得到值

函数指针 int (*p)(int x,int y); 形式必须和函数完全相同 p=fun fun(a, b); (*fun)(a, b); p(a, b); (*p)(a, b);

函数递归（函数自己调用自己的过程）

exit（0）：正常运行程序并退出程序；

exit（1）：非正常运行导致退出程序；

return：返回并退出函数【return 函数时，会执行再返回】

柯里化函数 function unCurried(x, y) { return x + y; } function curried(x) { return function(y) { return x + y; } } curried(1)(2) // 返回 3

TIPS：形参变量相当于被赋值，可以被改变 funciton(a,b){ a++,b++ }

Structure

Typedef

typedef <type-name> <new-type-name> Define a new type with typedef identif

// typedef int INT;

Definition

定义 （定义一个结构体指针时，定义时指向的空间为空---calloc）

不能使用结构体数组名直接访问成员，需要指针存储地址再进行操作

```
struct <struct-name> {
    int num;
```

```
    char name[20];
    char sex;
};              Define a structure directly. ( The structure name does not occupy memory, but is a data type similar to int, )
typedef struct   {
    int num;
    char name[20];
    char sex;
} <struct-name> ;              Define a structure directly with typedef keyword
struct <struct-name> {
    int num;
    char name[20];
    char sex;
} x,y;      Declare structural variables while defining structures.
struct <struct-name> {
    int num;
    char name[20];
    char sex;
} x,y;      Declare structural variables without defining structure name.
```

union 共同体，只能存在一个成员，取最大的成员所占内存空间（union 替换 struct 使用）

```
STUDENT *stu_1;   //定义结构体指针变量

//结构体指针初始化必须赋予一个有效地址，才能进行正常的操作
stu_1= (STUDENT *) malloc(sizeof(STUDENT *)); //为结构体指针 stu_1 申请内存空间

stu_1->student_id = 1234;
strcpy(stu1->student_name, "xiaoming");   //字符数组型结构体成员的赋值
stu_1->student_sex = 'M';
stu_1->student_math_score = 100;
```

List

```
strcpy(stu1.student_name, "xiaoming");   //字符数组型结构体成员的赋值
struct student stu[3];          //可以用的下标是 stu[0]--- stu[2]

//定义结构体数组的时候还可以同时进行初始化
struct student stu[ ]={ { },{ },{ } };
struct student stu[3] = {
{1001,"张三",1,18,"1 栋 1 单元",12,30,2000},
{1002,"李四",1,20,"2 栋 2 单元",11,15,1998},
{1003,"王五",1,22,"3 栋 3 单元",10,15,1996}
};
```

```
my_data data[]={
      { .name = "Peter" },
      { .name = "James" },
      { .name = "John" },
      { .name = "Mike" }
};

struct HLine{
    string name;
    color level;
    double price;
    string note;
};
HLine hLines[] = {{"SP", LD1, 1.07275,"test1"},  {"SD", LW1, 1.07275,"test2"}};
```
Constant expression required

C / Special Features

预处理

#include<name.h> 直接去系统配置的库中寻找

#include"name.h" 先在项目当前目录寻找头文件 再去系统配置的库中寻找

#define VV #define N 100 标识符宏名 用 N 表示 100（不能修改值，字符串中不包含宏）【计算时默认为 int，需要的到 double 类型需要强制转换】

#define M(a, b, c) a*b+c 带参数标识符宏名 先替换整个式子 再替换参数 ab

#undef M 清除宏名（不带参数）

宏定义

```
#ifndef cTest_Header_h
#define cTest_Header_h
//头文件内容
#endif
```

指针

int *p=&a; *p 取出地址对应的变量（通过地址可以直接改变变量的值）

NULL 空值（一个不被使用的地址，常为 0 也可以为其他地址） 不允许访问

动态储存（stdlib.h）

malloc(int size) 申请一个 size 字节大小的空间 不初始化 【返回空间地址-无类型需强制转换，失败返回 NULL】

memset(void *s, int x, size n); 将申请空间地址 s 的首 n 个字节设置为 x 值

calloc(int num, int size) 申请 num 个 size 字节大小的空间 初始化为 0 用数组访问 p[10]

free(地址) 释放申请空间

位运算

数用补码存储，补码运算（有符号时，正数首位 0 负数首位 1）

C / Library

输入输出

system("") 终止延迟程序执行 或 执行 cmd 命令

scanf("xxx%d", &num); 占位符+地址写入 %d %c %s %f %lf【必须按照"xxx%d"格式原样输入】

1. %c scanf("%c%c",&a, &b); 空格，回车，TAB 都将作为字符读入 多余数据会自动给下一个输入项（停留在缓存区）

scanf("%c %c",&a, &b);　　%c 前加空格，字符前的【空格，回车，TAB】将忽略不作输入

2. %s　scanf("%s", &a);　从地址 a 开始依次存入字符 （自动将最后的回车变'\0'）

中间有空格字符串结束，空格后的字符将作为下一个输入项（停留在缓存区）

3. scanf 中%f (float)　%lf (dobule)　printf 中%f（float&double）

printf("dog")　printf("%d%d", a, b)　不换行输出【无法打印二进制】　//printf("%d",a) == printf("%d", (int)a);

1. 占位符　%d（D）　%x(H)　%o(O)　%ld(long)　%hd(short)　%f（小数）

%c（字符）　%s（字符串）　%e（e 指数）　%E（E 指数）　%p（地址）　%%（百分号）

2. %-7.9f　-7 占位数但改变不了实际宽度（正左负右）　.9 为保留小数位　四舍五入

3. %s　必须对应一个地址　从地址开始输出，直到遇到 '\0' 输出结束

4. 双引号里除了%d　\n 以外　其他内容原样输出

gets(地址)　输入字符串　自动将　回车 变成 '\0'

puts(地址)　输出字符串　自动将　'\0'　变成 换行

getchar()　返回输入的字符

putchar(变量/值)　输出一个字符

无缓冲输入（conio.h ）　getche()　显示并返回输入字符　getch()直接返回输入字符

清空缓存区的命令　fflush(stdin)

注意：缓冲输入会自动换行

打开文件

FILE* p= fopen(文件绝对/相对路径, 打开方式)　失败返回 NULL　fclose(文件指针)【两次 open 同一个文件是不行的】

打开方式　read 读　write 写　append 追加　text 文本（可省略 t）　banary 二进制文件　+读和写

rt 只读打开文本文件　rb 只读打开二进制文件

wt 只写打开或建立一个文本文件　wt 只写打开或建立一个二进制文件　【w 打开文件时会直接先清空文件】

at 追加打开文本文件（在文件末尾写数据）　at 追加打开二进制文件（在文件末尾写数据）

rt+ rb+　wt+　wt+　at+　at+　读写打开【w 打开配合 fprintf 会出错】

fgetc(文件指针)　获取一个字符　【每读一个，光标后移】

fputc(字符，文件指针) 写一个字符　【每写一个，光标后移】

fgets(char *str，个数，文件指针)　从文件中读取 n-1 个字符 加上'\0'存入字符数组 str

fputs(char *str, 文件指针)　写入一个字符串

fscanf(fp, "%s %d %d %f\n", &a, &b, &c, &d);　从文件中读取指定格式数据 【每读一行，光标后移一行】

fprintf(fp, "%s %d %d %f\n", &a, &b, &c, &d);　写入文件一行字符串

feof(fp)　文件内光标后有字符 返回 0，没有返回非 0

rewind(fp)　将光标跳回文件头

数学（math.h）

sqrt(2.0)=1.414　根号 2.0

pow(3.14, 2)　3.14 的平方

字符串处理函数（string.h）

1. strlen(地址)　字符个数不算'\0'

2. strcpy(地址 a, 地址 b)　b 覆盖 a　(连'\0'也拷贝)

3. strcmp(地址 a, 地址 b)　依次用 a 减 b(ASCII 字符）结果不是零就自动结束　返回 1 或 0 或 -1【0 为完全相同】

4. strcat(地址 a,地址 b)　把 b 连接到 a 上（默认覆盖'\0'之后的内容）