简单 C 程序的说明

张晓平

武汉大学数学与统计学院

homepage: xpzhang.me

2018 年 3 月 6 日

目录

```c
// first.c:
#include<stdio.h>
int main(void)
/* A simple code */
{
   int num;
   num = 1;
   printf("I am a simple ");
   printf("computer.\n");
   printf("My lucky number  %d.\n", num);
   return 0;
}
```

#include与头文件

```
2 #include<stdio.h>
```

- 相当于在此处复制文件stdio.h的完整内容，以方便在多个程序间共享公用信息。
- #include语句是 C 预处理器指令的一个例子。通常，C 编译器在编译前要对源代码做一些准备工作，这称为预处理。
- 文件stdio.h包含了有关输入和输出函数的信息，以供编译器使用。

main 函数

3 `int main(void)`

C 程序至少包含一个函数，函数是 C 程序的基本模块。

表达式	含义
`( )`	函数名为 `main`
`int`	返回一整数
`void`	不接受任何参数

注：`main` 函数是任何 C 程序的唯一入口。

main 函数

main 函数有三种定义:

```
// Definition 1: NOT RECOMMENDED
void main() { /* ... */ }

// Definition 2
int main() { /* ... */ }

// Definition 3
int main(int argc, char* argv[]) { /* ... */ }
```

main 函数

考虑main 函数的两种定义，它们的差别是什么？

```
int main()
{
    /*   */
    return 0;
}
```

```
int main(void)
{
    /*   */
    return 0;
}
```

main 函数

考虑main 函数的两种定义，它们的差别是什么？

```
int main()
{
    /*  */
    return 0;
}

int main(void)
{
    /*  */
    return 0;
}
```

- 在 C++ 中，两种没有差别，完全一致。
- 在 C 中，两种定义都可以，但是第二种定义更好，因它清晰地表明 main() 在调用时不允许有参量。

main 函数

在 C 中，如果一个函数没有指定任何参量，就意味着该函数允许在调用时有任意多个参量或者无参量。

```
// Program 1 (Compiles and runs fine in C, but not in
 C++)
void fun() {  }
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

上述程序能编译和运行，但以下程序编译会失败。

```
// Program 2 (Fails in compilation in both C and C++)
void fun(void) {  }
int main(void)
{
    fun(10, "GfG", "GQ");
    return 0;
}
```

main 函数

- 在 C++ 中，以上两个程序在编译时都会失败，因为在 C++ 中，`fun()` 和 `fun(void)` 无差别。
- 在 C 中，`int main()` 和 `int main(void)`的差别在于，前者允许在调用时有任意多个参量，而后者在调用时不能有任何参量。尽管两者在大多数时候无任何差别，但在实践中更推荐使用 `int main(void)`。

main 函数

以下 C 程序的输出是什么？

```
#include <stdio.h>
int main()
{
  static int i = 5;
  if (--i){
    printf("%d ", i);
    main(10);
  }
}
```

```
#include <stdio.h>
int main(void)
{
  static int i = 5;
  if (--i){
    printf("%d ", i);
    main(10);
  }
}
```

注释

```
4 /* A simple code */
```

- `/* ... */` 之间的内容是程序注释。
- 注释可以让阅读者更容易理解程序。
- 注释可以放在任意位置，甚至和它要解释的语句在同一行。
- 一个较长的注释可以单独放一行，也可以是多行。
- `/* ... */` 之间的所有内容都会被编译器忽略。

注释


```
/* This is a C comment.*/

/* This comment is spread over
   two lines. */

/*
   You can do this, too.
*/

/* invalid comment
```

注释

C99 增加另一种风格的注释，使用 // 符号，该风格在 C++ 和 Java 中经常使用。

```
// Here is a comment confined to one line

int n; // Such comments can go here, too.
```

花括号，程序体和代码块


```
{
    ...
}
```

- C 函数使用花括号表示函数体的开始和结束。
- 花括号还可以用来把函数中的语句聚集到一个单元或代码块中。

声明

```
6    int num;
```

声明语句 (declaration statement)，做两件事情：

- 在内存中为变量 num 分配了空间。
- int 说明变量 num 的类型（整型）。

声明

```
6    int num;
```

声明语句 (declaration statement)，做两件事情：

- 在内存中为变量 num 分配了空间。
- int 说明变量 num 的类型（整型）。

注意：分号指明该行是 C 的一个语句。分号是语句的一部分。

声明

Ansi C 要求必须在一个代码块的开始处声明变量，在这之前不允许其他任何语句。

```
1 int main(void)
2 {
3    int n;
4    int m;
5    n = 5;
6    m = 3;
7    // other statements
8 }
```

声明

C99 遵循 C++ 的惯例，允许把声明放在代码块的任何位置。但是在首次使用变量之前仍必须先声明它。

```
1 int main(void)
2 {
3    int n;
4    n = 5;
5    // more statements
6    int m;
7    m = 3;
8    // other statements
9 }
```

声明

问题

- 什么是数据类型？
- 怎么给变量取名？
- 为什么必须对变量进行声明？

声明

1 数据类型

- C 可以处理多个数据种类，如整数、字符和浮点数。
- 把一个变量声明为整数类型、字符类型或浮点数类型，是计算机能正确存储、获取和解释该数据的基本前提。

2 取名

- “见其名只其意”。
- 若名字不能表达清楚，可以用注释解释变量所代表的意思。

3 命名规则：

- 只能使用字母、数字和下划线，且第一个字符不能为数字。
- 操作系统和 C 库通常使用以一个或两个下划线开始的名字，因此最好避免这种用法。
- 区分大小写，如 stars 不同于 Stars 或 STARS。

声明

表: 正确和错误的名字

✓	×
wiggles	$zj**
cat2	2cat
Hot_Dog	Hot-Dog
taxRate	tax Rate
_kcab	don't

声明变量的好处

- 把所有变量放在一起，可以让读者更容易掌握程序的内容。
- 在开始编写程序之前，考虑一下需要声明的变量会促使你做一些计划。
- 声明变量可以帮助避免程序中出现一类很难发现的细微错误，即变量名的错误拼写。
- 所有变量都需要声明，否则程序将不能编译。

赋值

```
7     num = 1;
```

这是一条赋值语句 (Assignment statement)，其含义：把值 1 赋给变量 num。

printf 函数

```
 8    printf("I am a simple ");
 9    printf("computer.\n");
10    printf("My lucky number  %d.\n", num);
```

printf 是一个标准输出函数，其信息由头文件 stdio.h 指定。

转义字符

转义字符通常用于代表难以表达或无法键入的字符，以 \ 开头。

转义字符	含义
\n	换行
\t	Tab 键
\b	退格
\'	单引号
\"	双引号
\\	反斜杠

格式化字符串

格式化字符串，也称占位符，用以指定输出项的数据类型和输出格式，以 % 开头。

占位符	含义
%d	用于输出十进制整数（实际长度）
%c	输出一个字符
%s	输出一个字符串
%f	以小数形式输出实数（整数部分全部输出，小数部分 6 位）

return语句

```
11    return 0;
```

带有返回值的 C 函数要求使用一个return语句，该语句包含关键字return，后面紧跟要返回的值。

使程序可读的技巧

- 变量命名时做到“见其名知其意”；
- 合理使用注释；
- 使用空行分隔一个函数的各个部分，如声明、操作等；
- 每条语句用一行。注意，C 允许把多条语句放在同一行或一条语句放多行。

使程序可读的技巧


```c
// mile_km.c: Convert 2 miles to kilometers
#include<stdio.h>
int main(void)
{
   float mile, km;
   mile = 2;
   km = 1.6 * mile;
   printf("%d mile = %d km\n", mile, km);
   printf("Yes, %d km\n", 1.6 * mile);
   return 0;
}
```

使程序可读的技巧

- 建议在程序开始处，用一个注释说明文件名和程序的作用，这对以后浏览或打印程序很有帮助。
- 多个声明

```
float mile, km;
```
⇔
```
float mile;
float km;
```

- 输出多个值
  - 第一个 printf 语句用了两个占位符：第一个%d 为 mile 占位，第二个%d 为 km 占位；圆括号中有三个参数，之间用逗号隔开。
  - 第二个 printf 语句说明输出的值可以是一个表达式。

多个函数

当程序比较复杂时，使用多个函数会可实现程序的模块化，使程序可读性更强。

多个函数

```c
#include<stdio.h>

float mile2km(float mile);

int main(void)
{
  float mile = 10, km;

  km = mile2km(mile);
  printf("%f mile = %f km\n",mile, km);

  return 0;
}

float mile2km(float mile)
{
  return 1.6 * mile;
}
```

多个函数

`mile2km()` 函数出现了三次：

1. 函数声明：通知编译器要用到该函数。
2. 函数调用
3. 函数定义

调试

找出以下程序中的错误。

```
1  #include<stdio.h>
2  int main(void)
3  (
4     int n, int n2, int n3;
5     /* 该程序含几个错误
6     n = 5; n2 = n * n;
7     n3 = n2 * n2;
8     printf("n = %d, n^2 = %d, n^3 = %d\n", n, n2, n3)
9     return 0;
10 )
```

语法错误

定义 (语法错误)

把正确的 C 符号放在了错误的位置。

语法错误

定义 (语法错误)

把正确的 C 符号放在了错误的位置。

1. 使用圆括号而不是花括号来包围函数体。
2. 声明方式应采用

```
int n, n2, n3;
```

或

```
int n;
int n2;
int n3;
```

3. 注释应该用“/* ... */”或“// ...”的形式。
4. printf语句最后漏掉了分号。

语法错误

问题

如何检测语法错误？

语法错误

问题

如何检测语法错误？

1. 在编译前看看源代码是否有明显的错误。
2. 查看编译器发现的错误。若有语法错误，编译时会报错，同时指出每一个错误的性质和位置。

语义错误

定义 (语义错误)

逻辑或含义上的错误。当语法没有错误，但结果不正确时，就是犯了语义错误。

语义错误

定义 (语义错误)

逻辑或含义上的错误。当语法没有错误，但结果不正确时，就是犯了语义错误。

```
n3 = n2 * n2;
```

本希望 n3 表示 n 的三次方，但求的却是 n 的四次方。

语义错误

定义 (语义错误)

逻辑或含义上的错误。当语法没有错误，但结果不正确时，就是犯了语义错误。

```
n3 = n2 * n2;
```

本希望 n3 表示 n 的三次方，但求的却是 n 的四次方。

语义错误编译器检测不到，它并没违法 C 规则。但编译器无法了解你的真正意图，只能靠你自己去发现这类错误。

语义错误

定义 (语义错误)

逻辑或含义上的错误。当语法没有错误，但结果不正确时，就是犯了语义错误。

```
n3 = n2 * n2;
```

本希望 n3 表示 n 的三次方，但求的却是 n 的四次方。

语义错误编译器检测不到，它并没违法 C 规则。但编译器无法了解你的真正意图，只能靠你自己去发现这类错误。

语义错误可以通过调试器来一步一步执行程序，来逐步跟踪和定位。

关键字

关键字是 C 中的特殊词汇，不能用它们来对变量或者函数命名。若试图把一个关键字作为变量名，编译器把它当做一个语法错误。

关键字

表: C 关键字

auto	enum	restrict	unsigned
break	extern	return	void
case	float	short	volatile
char	for	signed	while
const	goto	sizeof	_Bool
continue	if	static	_Complex
default	inline	struct	_Imaginary
do	int	switch	
double	long	typedef	
else	register	union	

关键字

表: C++ 关键字

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	