C 语言基础语法手册

January 9, 2026

1 预编译指令

1.1 概述

预编译指令是给编译器看的指令，通常以 # 开头，在编译前进行处理。

1.2 常用预编译指令

```c
#include <stdio.h>     // 引入标准输入输出头文件
#include "myheader.h" // 引入自定义头文件
#define PI 3.1415      // 定义常量（注意：末尾不要加分号）
```

Listing 1: 预编译指令示例

1.2.1 说明

- **#include**：用于引入头文件，使程序可以使用其中定义的函数和变量
- < > 和 " "：尖括号用于引入系统头文件，双引号用于引入用户自定义头文件
- **stdio.h**：全称"standard input/output"，包含 printf()、scanf()、fgets() 等输入输出函数
- **#define**：用于定义常量或宏，与 const 变量不同，它在编译前进行文本替换

> #define 定义的常量不需要加分号，因为它是在编译前进行简单的文本替换。

2 基本输入输出

2.1 printf() 函数

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

- 功能：格式化输出
- 参数：
  - `format`：格式字符串，包含要输出的文本和占位符
  - `...`：可变参数列表，按顺序替换占位符

2.2 scanf() 函数

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

- 功能：格式化输入
- 参数：与 printf() 类似，但需要传入变量的地址（使用 & 运算符）

2.3 基本示例

```c
#include <stdio.h>

int main() {
    int a;
    printf("请输入一个整数: ");
    scanf("%d", &a);          // &a 获取变量a的地址
    printf("a = %d\n", a);    // 输出变量a的值
    return 0;
}
```

Listing 2: 基本输入输出示例

3 占位符（格式说明符）

3.1 常用占位符

占位符	说明	示例
%d 或%i	十进制整数	`printf("%d", 10);`
%u	无符号十进制整数	`printf("%u", 20U);`
%x 或%X	十六进制整数（小写/大写）	`printf("%x", 255);` → ff
%o	八进制整数	`printf("%o", 8);` → 10
%f	浮点数（小数形式）	`printf("%f", 3.14);`
%e 或%E	浮点数（指数形式）	`printf("%e", 1000.0);` → 1.000000e+03
%g 或%G	自动选择%f 或%e（较简短形式）	`printf("%g", 1000.0);` → 1000
%c	单个字符	`printf("%c", 'A');`
%s	字符串	`printf("%s", "hello");`
%p	指针地址	`printf("%p", &a);`
%%		

Table 1: 常用占位符

3.2 占位符修饰符

```c
printf("%-10d", 123);      // 左对齐，宽度10
printf("%010d", 123);      // 用0填充，宽度10 → 0000000123
printf("%.2f", 3.14159);   // 保留2位小数 → 3.14
printf("d", 5, 10); // 宽度由参数指定（5）→ "  10"
```

Listing 3: 占位符修饰符示例

4 转义字符

用于表示特殊字符的符号，以反斜杠 \开头。

4.1 常见转义字符

转义字符	说明
\n	换行（光标移到下一行开头）
\t	水平制表符（通常 4 个空格）
\r	回车（光标移到当前行开头）
\b	退格（删除前一字符）
\f	换页符
\a	警报（系统提示音）
\\	反斜杠字符
\’	单引号字符
\”	双引号字符
\?	问号字符
\0	空字符（字符串结束标志）

Table 2: 常见转义字符

4.2 示例

```
printf("Hello\tWorld!\n");     // Hello    World!（然后换行）
printf("路径：C:\\Program Files\\\n");  // 路径：C:\Program Files\
```

Listing 4: 转义字符示例

5 运算符

5.1 算术运算符

```
+  // 加法
-  // 减法
*  // 乘法
/  // 除法
%  // 取模（求余数）
++ // 自增（前缀/后缀）
-- // 自减（前缀/后缀）
```

5.2 关系运算符

```
== // 等于
!= // 不等于
>  // 大于
<  // 小于
>= // 大于等于
<= // 小于等于
```

5.3 逻辑运算符

```
&& // 逻辑与（AND）
|| // 逻辑或（OR）
!  // 逻辑非（NOT）
```

5.4 位运算符

```
&  // 按位与
|  // 按位或
^  // 按位异或
~  // 按位取反
<< // 左移
>> // 右移
```

5.5 地址相关运算符

```
&  // 取地址运算符：获取变量地址
*  // 解引用运算符：通过指针访问值
```

5.6 示例

```
int a = 10;
int *ptr = &a;      // ptr存储a的地址
printf("%d\n", *ptr);  // 输出10（通过指针访问a的值）
```

Listing 5: 运算符示例

6 字符输入函数

6.1 getchar() 函数

```
int getchar(void);
```

- 从标准输入读取单个字符
- 返回读取字符的 ASCII 值（类型为 `int`，可存储 EOF）
- **EOF**（End Of File）：值为-1，表示输入结束（Windows: Ctrl+Z，Linux: Ctrl+D）

6.2 清空输入缓冲区示例

```c
#include <stdio.h>

int main() {
    int ch;
    char name[20];
    char sex;

    printf("请输入姓名: ");
    scanf("%s", name);

    // 清空缓冲区中的换行符等残留字符
    while ((ch = getchar()) != '\n' && ch != EOF);

    printf("请输入性别(F/M): ");
    scanf("%c", &sex);

    printf("姓名: %s, 性别: %c\n", name, sex);
    return 0;
}
```

Listing 6: 清空缓冲区示例

7 字符串输入函数

7.1 fgets() 函数

```c
char *fgets(char *str, int n, FILE *stream);
```

- 从指定流读取字符串
- 最多读取 n-1 个字符，自动添加 \0
- 保留换行符 \n（如果读取到）
- 相比 gets() 更安全，避免缓冲区溢出

7.2 示例

```c
#include <stdio.h>

int main() {
    char buffer[20];
    printf("请输入字符串: ");
    fgets(buffer, 20, stdin);  // 最多读取19个字符
    printf("读取内容: %s", buffer);
    return 0;
}
```

Listing 7: fgets() 示例

7.3 注意事项

- 读取字符数限制为 n-1
- 会存储换行符 \n
- 不会清空目标数组的未使用部分
- 输入过长时，剩余字符留在输入缓冲区
- 成功时返回 str，失败或到达文件末尾返回 NULL

使用 fgets() 时，如果输入的字符数超过 n-1，剩余字符会留在输入缓冲区中，可能会影响后续的输入操作。

8 数组

8.1 数组声明与初始化

```c
int arr[5];                     // 声明长度为5的整型数组
int arr[3] = {1, 2, 3};         // 声明并初始化
int arr[] = {1, 2, 3, 4, 5};    // 自动推断长度
```

Listing 8: 数组示例

8.2 数组访问

```c
int a[3] = {1, 2, 3};
printf("%d\n", a[0]);     // 输出第一个元素：1
printf("%d\n", a[1]);     // 输出第二个元素：2
printf("%d\n", a[2]);     // 输出第三个元素：3
```

Listing 9: 数组访问示例

8.3 重要特性

- 数组名是数组首元素的地址（指针）
- a[0] 等价于 *(a + 0) 等价于 *(0 + a) 等价于 0[a]
- 索引从 0 开始

9 字符串

9.1 字符串本质

- 字符串是字符数组，以空字符 \0 结尾
- 单引号 ' ' 表示单个字符
- 双引号 " " 表示字符串（自动添加 \0）

9.2 示例

```c
char ch = 'A';              // 单个字符
char str1[] = "Hello";      // 字符串，实际长度为6（包含\0）
char str2[] = {'H', 'e', 'l', 'l', 'o', '\0'};  // 等价写法
```

Listing 10: 字符串示例

9.3 字符串与指针

```c
char *str = "Hello";        // 字符串字面量，str指向字符串常量区
char arr[] = "World";       // 字符数组，可修改
```

Listing 11: 字符串与指针

10 指针基础

10.1 指针概念

- 指针是存储内存地址的变量
- 可以指向其他变量，也可以进行算术运算

10.2 指针操作

```c
int a = 10;
int *ptr = &a;      // ptr指向a的地址

printf("a的值: %d\n", a);          // 10
printf("a的地址: %p\n", &a);       // 地址值
printf("ptr的值: %p\n", ptr);      // 与&a相同
printf("*ptr的值: %d\n", *ptr);    // 10（解引用）
```

Listing 12: 指针操作示例

10.3 指针与数组关系

```c
int arr[3] = {1, 2, 3};
int *ptr = arr;             // 等价于 int *ptr = &arr[0];

printf("%d\n", arr[1]);     // 2
printf("%d\n", *(arr+1));   // 2
printf("%d\n", ptr[1]);     // 2
printf("%d\n", *(ptr+1));   // 2
```

Listing 13: 指针与数组关系

11 综合示例

```c
#include <stdio.h>
#define PI 3.14159

int main() {
    // 基本类型
    int num = 100;
    float f = 3.14;
    char ch = 'A';
    char str[] = "Hello World";

    // 输出演示
    printf("整数: %d, 浮点数: %.2f\n", num, f);
    printf("字符: %c, 字符串: %s\n", ch, str);
    printf("十六进制: %x, 八进制: %o\n", num, num);

    // 指针演示
    int *p = &num;
    printf("num的地址: %p, 通过指针访问值: %d\n", p, *p);

    // 数组演示
    int arr[] = {10, 20, 30};
    printf("数组元素: %d, %d, %d\n", arr[0], arr[1], arr[2]);

    // 转义字符演示
    printf("第一行\n第二行\t缩进\n");

    return 0;
}
```

Listing 14: 综合示例

12 重要提示

- 格式匹配：占位符必须与参数类型匹配，否则可能输出错误或程序崩溃
- 地址传递：scanf() 需要变量的地址（& 运算符），但数组名本身就是地址
- 缓冲区问题：混合使用 scanf() 和 fgets()/getchar() 时注意清空缓冲区
- 字符串结束符：处理字符串时确保以 \0 结尾
- 指针安全：未初始化的指针不要解引用，避免野指针
- 数组边界：访问数组时不要越界
- 内存管理：动态分配的内存要及时释放

12.1 常见错误

```c
// 错误：未初始化的指针
int *ptr;
*ptr = 10;  // 错误！ptr未初始化

// 错误：数组越界
int arr[5];
arr[5] = 10;  // 错误！索引从0到4

// 错误：格式不匹配
float f = 3.14;
printf("%d\n", f);  // 错误！应该用%f
```

Listing 15: 常见错误示例

12.2 最佳实践

1. 始终检查输入函数的返回值
2. 使用 fgets() 代替 gets() 以避免缓冲区溢出
3. 使用 const 修饰不应该被修改的指针参数
4. 为指针分配内存后检查是否分配成功
5. 释放内存后将指针设为 NULL

13 练习题目

13.1 基础题

1. 编写一个程序，从用户输入读取一个整数并输出其平方。
2. 编写一个程序，读取两个整数并交换它们的值。
3. 编写一个程序，读取一个字符串并输出其长度。

13.2 进阶题

1. 编写一个程序，实现简单的计算器功能（加减乘除）。
2. 编写一个程序，统计输入文本中的字符、单词和行数。
3. 编写一个程序，实现字符串的逆序输出。

这份手册涵盖了 C 语言基础语法的核心内容，可作为学习和参考的指南。建议结合实际编程练习来加深理解。