C 语言程序设计

卓能文

智能科技学院

2024-02-21

环境安装

1. msys2: https://www.msys2.org/ https://github.com/msys2/msys2-installer/releases/download/2024-01-13/msys2-x86_64-20240113.exe
2. visual studio code: https://code.visualstudio.com/ 下载并安装合适版本

编辑、编译、运行

```c
#include <stdio.h>

int main() {
    printf("Hello, World!");
    return 0;
}
```

注释

在 C 语言中，注释语句用于对代码进行解释和说明，可以提高代码的可读性和可维护性，不会被编译器执行。C 语言有两种注释方式：

单行注释：使用//开头，后面的内容直到该行结束都被视为注释。

```
// 这是一个单行注释
int a = 10; // 定义一个整型变量 a 并赋值为 10
```

多行注释：使用/*开头，*/结尾，中间的内容都被视为注释。

```
/*
这是一个多行注释
可以跨越多行
```

注释

```
*/
int b = 20;
```

变量定义

在 C 语言中，变量定义是指为变量分配内存空间并指定其数据类型。变量定义的基本语法如下：

数据类型 变量名;

例如，定义一个整型变量 age，可以这样写：

```
int age;
```

变量赋值

在 C 语言中，变量赋值是指将一个值赋给变量。变量赋值的基本语法如下： 变量名 = 值;

例如，将变量 age 的值赋为 20，可以这样写：

```
age = 20;
```

变量输出

在 C 语言中，输出变量可以使用 printf()函数。例如：

```c
#include <stdio.h>

int main() {
    int num = 10;
    printf("num 的值为: %d", num);
    return 0;
}
```

其中，%d 表示输出整数类型的变量，如果需要输出其他类型的变量，可以使用相应的格式控制符。

条件语句

在 C 语言中，条件语句用于根据指定的条件执行不同的代码块。条件语句的基本语法如下：

if (表达式) {}

例如，定义一个整型变量 age，并将其赋值为 20，然后使用条件语句判断 age 是否大于 18：

```
int age = 20;

if (age > 18) {
    printf("你已经成年了！\n");
}
```

条件语句

在 C 语言中，条件语句通常使用 if、else if 和 else 关键字。例如：

```c
#include <stdio.h>

int main() {
    int num = 10;
    if (num > 5) {
        printf("num 大于 5");
    } else if (num == 5) {
        printf("num 等于 5");
    } else {
        printf("num 小于 5");
    }
```

条件语句

```
    return 0;
}
```

在这个例子中，程序首先判断 num 是否大于 5，如果是则输出“num 大于 5”，否则继续判断 num 是否等于 5，如果是则输出“num 等于 5”，否则输出“num 小于 5”。

while 语句

在 C 语言中，循环语句用于重复执行一段代码。循环语句的基本语法如下：

while (表达式) {}

例如，定义一个整型变量 i，并将其赋值为 0，然后使用循环语句重复执行代码块，直到 i 大于等于 10：

```c
int i = 0;

while (i < 10) {
    printf("i 的值为: %d\n", i);
```

while 语句

```
        i++;
    }
```

for 语句

在 C 语言中，for 语句用于循环执行一段代码。它的基本语法如下：

```
for (初始化表达式;  条件表达式;  更新表达式) {
    // 循环体
}
```

其中，初始化表达式用于设置循环变量的初始值；条件表达式用于判断是否继续执行循环体；更新表达式用于更新循环变量的值。例如，下面的代码使用 for 语句输出 1 到 10 的数字：

```
#include <stdio.h>

int main() {
```

for 语句

```
    for (int i = 1; i <= 10; i++) {
        printf("%d ", i);
    }
    return 0;
}
```

在这个例子中，i 是循环变量，初始值为 1，每次循环后增加 1，直到 i 大于 10 时停止循环。

break 语句

在 C 语言中，break 语句用于跳出循环或者 switch 语句。当程序执行到 break 语句时，会立即退出当前所在的循环或 switch 语句，继续执行后面的代码。例如：

```c
#include <stdio.h>

int main() {
    for (int i = 0; i < 10; i++) {
        if (i == 5) {
            break;
        }
        printf("%d ", i);
    }
```

break 语句

```
    return 0;
}
```

这个例子中，程序使用 for 语句输出 0 到 9 的数字。当 i 等于 5 时，程序执行到 break 语句，立即退出循环，不再输出数字。因此，程序只输出 0 到 4 的数字。

continue 语句

在 C 语言中，continue 语句用于跳过当前循环的剩余部分，直接进入下一次循环。当程序执行到 continue 语句时，会立即跳过当前循环体中 continue 后面的代码，并开始下一次循环。例如：

```c
#include <stdio.h>

int main() {
    for (int i = 0; i < 10; i++) {
        if (i % 2 == 0) {
            continue;
        }
        printf("%d ", i);
    }
```

continue 语句

```
    return 0;
}
```

这个例子中，程序使用 for 语句输出 0 到 9 的数字。当 i 是偶数时，程序执行到 continue 语句，跳过本次循环的剩余部分，直接进入下一次循环。因此，程序只输出奇数 0 到 9 的数字。

函数

在 C 语言中，函数是指一组语句的集合，用于完成特定的任务。函数的基本语法如下：

返回值类型 函数名(参数列表) { return 返回值; }

例如，定义一个函数 add，用于计算两个整数的和，可以这样写：

```
int add(int a, int b) {
    int sum = a + b;
    return sum;
}
```

数组

在 C 语言中，数组是指一组相同数据类型的元素的集合。数组的基本语法如下：

数据类型 数组名[元素个数];

例如，定义一个包含 5 个整型元素的数组 numbers，可以这样写：

```
int numbers[5];
```

指针

在 C 语言中，指针是指向内存地址的变量。指针的基本语法如下：

数据类型 *变量名;

例如，定义一个指向整型变量的指针 p，可以这样写：

```
int num = 10;
int *p = &num;
```

字符串

在 C 语言中，字符串是指一组由空字符'0'分隔的字符。字符串的基本语法如下：

char 数组名[字符个数 + 1];

例如，定义一个包含 5 个字符的字符串 str，可以这样写：

```
char str[6] = "Hello";
```

结构体

在 C 语言中，结构体是指一组不同数据类型的元素的集合。结构体的基本语法如下：

struct 结构体名 { 成员 1 类型 成员 1 名; 成员 2 类型 成员 2 名; ... };

例如，定义一个包含两个整型变量的结构体 Person，可以这样写：

```
struct Person {
    int age;
    char name[20];
};
```

文件操作

在 C 语言中，文件操作是指对文件进行读取、写入、删除等操作。文件操作的基本语法如下：

```
FILE *fopen(char *filename, char *mode);
```

例如，打开一个名为"example.txt"的文件，可以这样写：

```
FILE *file = fopen("example.txt", "r");
```

指针和数组

在 C 语言中，指针和数组是两种不同的数据类型，但它们之间存在一些相似之处。指针和数组都可以用来存储多个相同数据类型的元素。指针和数组都可以通过下标来访问其中的元素。

例如，定义一个包含 5 个整型元素的数组 numbers，可以这样写：

```
int numbers[5];
```

定义一个指向整型变量的指针 p，可以这样写：

```
int num = 10;
int *p = &num;
```

函数和指针

在 C 语言中，函数和指针是两种不同的概念，但它们之间存在一些相似之处。函数可以作为参数传递给其他函数，也可以作为返回值返回。指针可以指向函数，也可以通过函数指针调用函数。

例如，定义一个函数 add，用于计算两个整数的和，可以这样写：

```
int add(int a, int b) {
    return a + b;
}
```

定义一个指向函数的指针 p，可以这样写：

```
int (*p)(int, int);
```

函数和数组

在 C 语言中，函数和数组是两种不同的概念，但它们之间存在一些相似之处。函数可以作为参数传递给其他函数，也可以作为返回值返回。数组可以作为参数传递给其他函数，也可以作为返回值返回。

例如，定义一个函数 printArray，用于打印数组中的元素，可以这样写：

```c
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

函数和数组

定义一个函数 sumArray，用于计算数组中的元素之和，可以这样写：

```
int sumArray(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}
```