姓名：席崇援

学习任务：

了解c语言

1.理解头文件，源文件，主函数的定义和使用；

2.能打印“Hello World!”;

3.数据类型（基本数据类型，枚举类型，void，派生类型等等），并找到类型转换；

4.变量，整型，浮点型的定义和使用；

5.常量的定义和使用，变量和常量的区别；

6.存储类；

7.运算符（重点注意++a和a++的区别以及/和%的区别）。

8.判断语句；

9.循环语句。

完成情况：

➤!!!存储类没弄

大象喝水

```c
#include<stdio.h>
#include<math.h>
int main()
{
        int H,R;
        double pi=3.14;
        printf("请输入小圆桶的深度H,底面半径R\n");
        scanf("%d %d",&H,&R);
        double V=pi*R*R*H;
        int T=ceil(20000/V);//向上取整
        if (V<=20000) {
                printf("大象至少要喝%d桶水",T);
        } else {
                printf("这样的一桶水就够大象喝了");
        }
        return 0;
}
```

笔记：

---

①头文件通常以 `.h` 为扩展名，`#include` 预处理指令来引入头文件；

②源文件是以 `.c` 为扩展名的文件，其中包含了程序的主要逻辑实现，一个项目可能包含多个源文件。

③主函数：`main` 函数是C程序执行的起点。每个C程序都必须有一个 `main` 函数。

定义头文件

```c
#ifndef SUM_H
#define SUM_H

int add(int a, int b);

#endif // SUM_H
```

源文件

```
#include "sum.h"

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

打印“Hello World!”

```
#include <stdio.h>
int main()
{
   printf("Hello, World!");
   return 0;
}
```

基本数据类型

- **整型**（`int`，`short`，`long`，`long long`，`unsigned int`，`unsigned long`，等）
  - **`int`**：一般整型，通常占用32位（4字节）。
  - **`short`**：短整型，通常占用16位（2字节）。
  - **`long`**：长整型，通常占用32位（4字节）或64位（8字节）。
  - **`long long`**：非常长的整型，通常占用64位（8字节）。
- **浮点型**（`float`，`double`，`long double`）
  - **`float`**：单精度浮点型，通常占用32位（4字节）。
  - **`double`**：双精度浮点型，通常占用64位（8字节）。
  - **`long double`**：扩展精度浮点型，通常占用80位（10字节）或更多。

eg:

```c
#include <stdio.h>
int main() {
    // 整型变量
    int a = 10;  // 声明并初始化一个整型变量 a，其值为 10
    short b = 20;  // 声明并初始化一个短整型变量 b，其值为 20
    long c = 30L;  // 声明并初始化一个长整型变量 c，其值为 30
    long long d = 40LL;  // 声明并初始化一个非常长的整型变量 d，其值为 40

    // 浮点型变量
    float e = 3.14f;  // 声明并初始化一个单精度浮点型变量 e，其值为 3.14
    double f = 2.71828;  // 声明并初始化一个双精度浮点型变量 f，其值为 2.71828
    long double g = 1.4142135623730951;  // 声明并初始化一个扩展精度浮点型变量 g，其值为

    // 输出变量的值
    printf("a = %d\n", a);  // 输出整型变量 a 的值
    printf("b = %hd\n", b);  // 输出短整型变量 b 的值，%hd 用于输出 short 类型
    printf("c = %ld\n", c);  // 输出长整型变量 c 的值，%ld 用于输出 long 类型
    printf("d = %lld\n", d);  // 输出非常长的整型变量 d 的值，%lld 用于输出 long long 类
    printf("e = %.2f\n", e);  // 输出单精度浮点型变量 e 的值，%.2f 控制小数点后两位
    printf("f = %.6f\n", f);  // 输出双精度浮点型变量 f 的值，%.6f 控制小数点后六位
    printf("g = %.10Lf\n", g);  // 输出扩展精度浮点型变量 g 的值，%.10Lf 控制小数点后十位

    // 修改变量的值
    a = 20;  // 将整型变量 a 的值修改为 20
    b = 30;  // 将短整型变量 b 的值修改为 30
    c = 40L;  // 将长整型变量 c 的值修改为 40
    d = 50LL;  // 将非常长的整型变量 d 的值修改为 50

    // 再次输出变量的值
    printf("a = %d\n", a);  // 输出整型变量 a 的新值
    printf("b = %hd\n", b);  // 输出短整型变量 b 的新值
    printf("c = %ld\n", c);  // 输出长整型变量 c 的新值
    printf("d = %lld\n", d);  // 输出非常长的整型变量 d 的新值

    return 0;  // 返回 0，表示程序正常结束
}
```

- **字符型**（`char`）

枚举类型

枚举类型用于定义一组命名的整数常量。

```
enum Color { Red, Green, Blue };
eg:
#include <stdio.h>

// 定义枚举类型
enum Color {
    Red,
    Green,
    Blue
};

int main() {
    // 创建枚举变量并初始化
    enum Color color = Red;

    // 输出枚举变量的值
    printf("Color: %d\n", color);

    // 更改枚举变量的值
    color = Green;
    printf("Color: %d\n", color);

    // 更改枚举变量的值
    color = Blue;
    printf("Color: %d\n", color);

    return 0;
}
```

`void` 类型

`void` 类型表示“无类型”，常用于函数返回值或指针类型。

```
void myFunction();
void *myPointer;
```

派生类型

派生类型是从基本类型派生出来的类型，包括数组、指针、结构体、联合体等。

- **数组**

```
int arr[10];
```

- **指针**

```
int *p;
```

- **结构体**

```
struct Person {
    char name[50];
    int age;
};
```

- **联合体**

```
union Data {
    int i;
    float f;
};
```

类型转换

类型转换指将一种数据类型转换为另一种数据类型。C语言支持两种类型的类型转换：

- **隐式类型转换**
- **显式类型转换**

隐式类型转换

隐式类型转换是编译器自动完成的类型转换。

```
int a = 5;
float b = a;  // 自动将 int 转换为 float
```

显式类型转换

显式类型转换是由程序员明确指定的类型转换，通常使用类型强制转换语法。

```
int a = 5;
float b = (float) a;  // 显式将 int 转换为 float
```

运算符(主要)

算术运算符

- **加法**（`+`）
- **减法**（`-`）
- **乘法**（`*`）
- **除法**（`/`）
- **取模**（`%`）

关系运算符(返回一个布尔结果)

- **等于**（`==`）
- **不等于**（`!=`）
- **大于**（`>`）
- **小于**（`<`）
- **大于等于**（`>=`）
- **小于等于**（`<=`）

逻辑运算符(返回一个布尔结果)

- **与**（`&&`）
- **或**（`||`）
- **非**（`!`）

位运算符(对二进制位进行操作)

- **按位与**（`&`）
- **按位或**（`|`）
- **按位异或**（`^`）-如果两个相应的位相同，则结果为 0；如果不同，则结果为 1
- **按位取反**（`~`）-每个二进制位 0 变为 1，每个二进制位 1 变为 0
- **左移**（`<<`）
- **右移**（`>>`）

```
a = 60 = 0011 1100 (二进制)
b = 13 = 0000 1101 (二进制)
a ^ b = 0011 1100
      ^ 0000 1101
      ----------
        0011 0001 = 49 (十进制)

value = 8 = 0000 1000 (二进制, 8位)
~value = ~0000 1000
       = 1111 0111 (二进制)

int a = 5;    // 二进制表示: 0101
int b = a << 1; // 左移1位, 结果: 01010 (即十进制的 10)
int c = a << 2; // 左移2位, 结果: 010100 (即十进制的 20)

int a = 10;   // 二进制表示: 01010
int b = a >> 1; // 右移1位, 结果: 00101 (即十进制的 5)
int c = a >> 2; // 右移2位, 结果: 00010 (即十进制的 2)
```

判断语句

用于根据条件来决定程序的执行路径。

1. **if 语句**
2. **if-else 语句**
3. **if-else-if 语句**
4. **switch 语句**

1. if 语句

`if` 语句用于根据一个条件来决定是否执行某个代码块。

```c
#include <stdio.h>

int main() {
    int a = 10;

    if (a > 5) {
        printf("a is greater than 5\n");
    }

    return 0;
}
```

2. if-else 语句

`if-else` 语句用于根据一个条件来决定执行两个不同的代码块之一。

```c
#include <stdio.h>

int main() {
    int a = 10;

    if (a > 5) {
        printf("a is greater than 5\n");
    } else {
        printf("a is less than or equal to 5\n");
    }

    return 0;
}
```

3. if-else-if 语句

`if-else-if` 语句用于根据多个条件来决定执行哪个代码块。

```c
#include <stdio.h>

int main() {
    int a = 10;

    if (a > 10) {
        printf("a is greater than 10\n");
    } else if (a == 10) {
        printf("a is equal to 10\n");
    } else {
        printf("a is less than 10\n");
    }

    return 0;
}
```

4. switch 语句

`switch` 语句用于根据一个表达式的值来决定执行哪个代码块。

```c
#include <stdio.h>

int main() {
    int day = 3;

    switch (day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        case 3:
            printf("Wednesday\n");
            break;
        default:
            printf("Other day\n");
    }

    return 0;
}
```

9. 循环语句

循环语句用于重复执行一段代码，直到满足某个条件为止。C语言中主要有以下几种循环语句：

1. **for 循环**
2. **while 循环**
3. **do-while 循环**

    - **for 循环==while 循环**

1. for 循环

`for` 循环用于在已知次数的情况下重复执行一段代码。

```
#include <stdio.h>

int main() {
    for (int i = 1; i <= 5; i++) {//初始化表达式; 循环条件; 更新表达式
        printf("%d\n", i);
    }

    return 0;
}
```

2. while 循环

`while` 循环用于在未知次数的情况下重复执行一段代码，直到条件不再满足。

```
#include <stdio.h>

int main() {
    int i = 1;

    while (i <= 5) {
        printf("%d\n", i);
        i++;
    }

    return 0;
}
```

3. do-while 循环

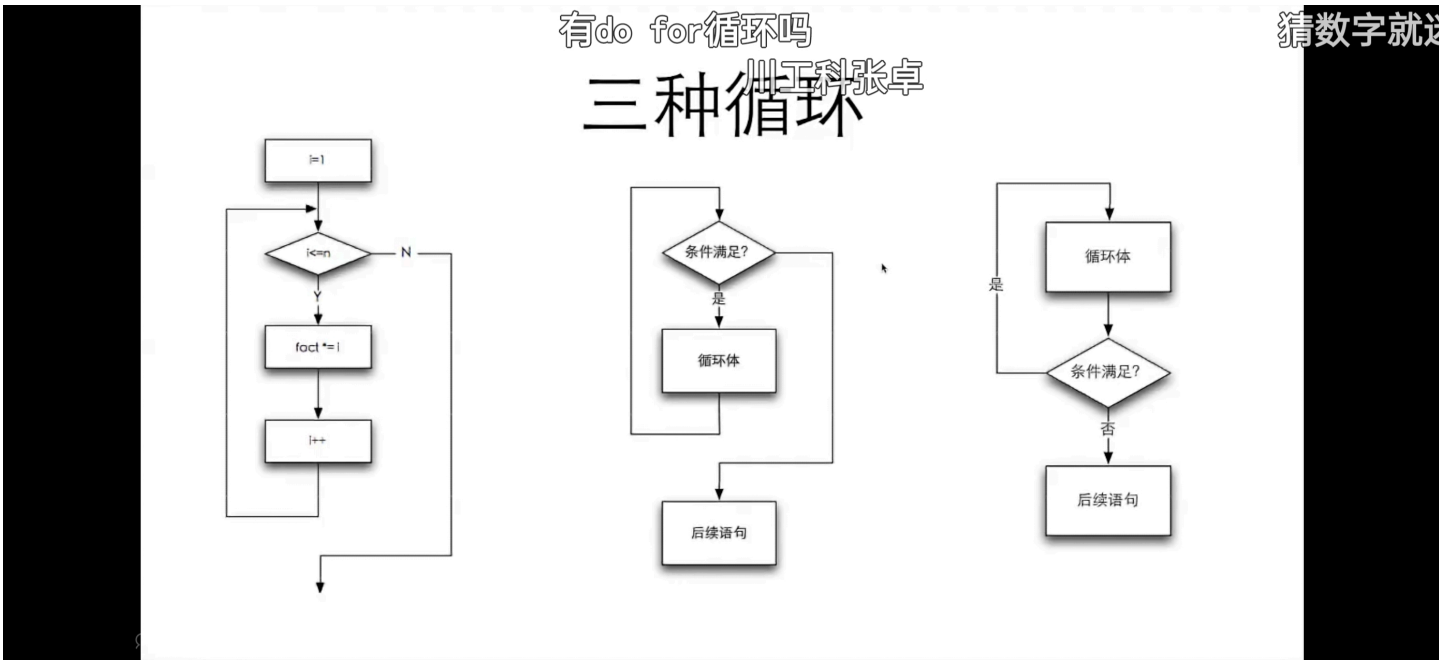
`do-while` 循环至少会执行一次循环体，然后根据条件决定是否继续执行。

```c
#include <stdio.h>

int main() {
    int i = 1;

    do {
        printf("%d\n", i);
        i++;
    } while (i <= 5);

    return 0;
}
```



格式化输出符号

%d 十进制有符号整数

%u 十进制无符号整数

%f 浮点数

%s 字符串

%c 单个字符

%p 指针的值

%e 指数形式的浮点数

%x, %X 无符号以十六进制表示的整数

%o 无符号以八进制表示的整数

%g 把输出的值按照 %e 或者 %f 类型中输出长度较小的方式输出

%p 输出地址符

%lu 32位无符号整数

%llu 64位无符号整数

%2f是把float的所有位数输出2位,包括小数点,如果不组2位,补0,如果超过2位,按照实际输出

%.2f是float后的小数只输出两位。

%d=int

%ld=long

%lld=long long