C语言第一课

90分钟：从第一行代码，到变量、运算、循环和函数

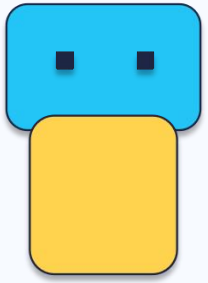
今天不背概念：每讲一点，就马上验证。

目标：看懂5类基础积木，完成3个程序练习。

关键词：数据类型、运算符、表达式、语句、循环、函数。

今天会看懂这句

```
printf("Hello, C!");
```



编译器

今天的学习地图

3分钟

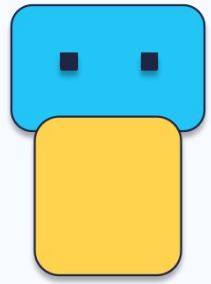
把C语言第一课拆成几个明确任务，学完能继续往下走。

引入	→	数据类型	→	运算/语句	→	循环	→	函数
电脑听指令		不同数据不同存法		让数据动起来		重复做事		把代码变成工具

每个任务都有：一个直观解释、一段代码、一个练习。

第11页安排5分钟休息。

最后会看到函数：把常用代码打包成工具。



编译器

引入：电脑到底怎么执行指令？

5分钟

电脑很快，但不会猜。程序员要把步骤说清楚。

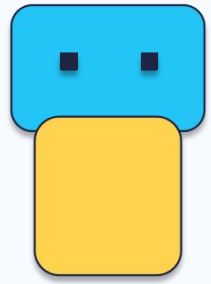
生活比喻：写一份让别人完全照做的实验步骤。
不能只写“加点水”，要写“加入200毫升水”。
C语言就是写给电脑的精确执行步骤。
课堂互动：让同学用准确指令指挥老师从讲台走到门口。

快速判断

哪条指令更像程序？

A. 去那边

B. 向前走3步，右转，再走2步



编译器

结论：电脑不是聪明地猜，而是严格地执行。

第一个C程序：先看骨架

5分钟

先不用全部背下来，先知道每块大概做什么。

#include：借工具箱，printf就在里面。

main：程序从这里开始。

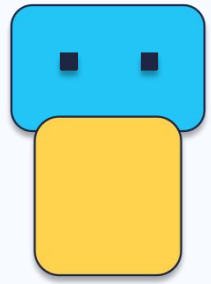
printf：把文字打印到屏幕。

return 0：告诉系统“我正常结束”。

代码示例

```
#include <stdio.h>

   int main() {
printf("Hello, C!\n");
       return 0;
         }
```



编译器

提醒：英文符号很重要，分号、双引号、大括号都不能随便换。

数据类型：不同数据要用不同存法

6分钟

变量是有名字的存储位置；类型决定它适合保存什么。

为什么要分类型？因为电脑要提前知道

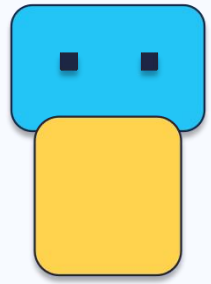
。

字、

int	double	char	char[]
存整数	存小数	存单字符	存一段文字
12, -5, 100	3.14, 1.68	'A', '?'	"XiaoMing"

最小代码

```
int age = 12;
double height = 1.55;
char level = 'A';
char name[] = "Ming";
```



编译器

直觉：类型不是为了难为你，而是为了让电脑明确地存储和计算。

常用类型和printf暗号

6分钟

printf要用占位符，告诉电脑用什么方式打印。

int → %d：整数

double → %.2f：保留2位小数

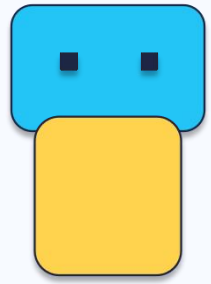
char → %c：单个字符

char[] → %s：一串文字

代码示例

```
int age = 12;
double price = 3.5;
char grade = 'A';
char name[] = "XiaoMing";

printf("%s今年%d岁\n", name, age);
printf("价格%.2f元，等级%c\n", price,
grade);
```



编译器

记忆方式：整数d，小数f，字符c，字符串s。

练习1：个人信息卡

7分钟

把自己的信息放进变量，再打印出来。

扫码打开在线C编译器

任务卡

任务：改变量值，打印一张信息卡。

必须有：姓名、年龄、喜欢的数字。

进阶：再加一个小数，例如身高或跑步用时。

参考代码

```
char name[] = "小明";
int age = 12;
int lucky = 7;
double height = 1.52;

printf("我是%s，今年%d岁\n", name,
age);
printf("幸运数字是%d，身高%.2f米\n",
lucky, height);
```

编译器

成功标准：能改变量，能解释每个占位符在打印什么。

运算符：让数据参与计算和判断

6分钟

运算符决定电脑怎样处理变量和值。

算术：+ - * / %

关系：> < >= <= == !=

逻辑：&& || !

赋值：= += -= *= /=

重点例子

10 / 3 = 3

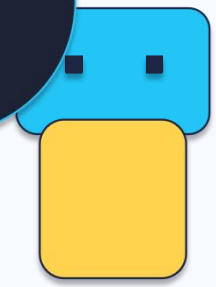
10 % 3 = 1

因为int除法会丢掉小数部分。

%特别适合判断奇偶数。

代码示例

```
int a = 10;
int b = 3;
printf("%d\n", a / b);  // 3
printf("%d\n", a % b);  // 1
printf("%d\n", a % 2);  // 0表示偶数
```



编译器

提醒：=是“放进去”，==才是“是否相等”。

表达式和语句：一行命令怎么读？

5分钟

表达式会算出一个值；语句是让电脑执行的一句话。

表达式：3 + 4、age + 5、score >= 60。

语句：以分号结尾的一条命令。

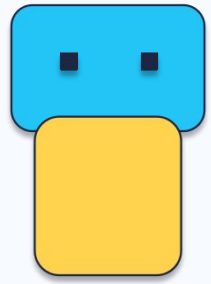
程序通常按顺序从上往下执行。

括号可以改变计算顺序，像数学课一样。

代码示例

```
    int score = 80;                 // 语句
int next = score + 10;          // 表达式 score
                        + 10

if (next >= 90) {               // 条件也是表达
                            式
                printf("优秀\n");
                        }
```



编译器

口诀：表达式算结果，语句做动作。

练习2：购物计算器

8分钟

用运算符和表达式完成真实计算。

扫码打开在线C编译器

任务卡

题目：铅笔2元，橡皮3元，买4支铅笔和2块橡皮，一共多少钱？

再算：如果带20元，还剩多少钱？

挑战：把数量改成变量，随便换数字也能算。

参考代码

```
int pencilPrice = 2;
int eraserPrice = 3;
int pencilCount = 4;
int eraserCount = 2;
   int money = 20;

int total = pencilPrice *
pencilCount + eraserPrice *
        eraserCount;
printf("一共%d元，还剩%d元\n", total,
        money - total);
```

编译器

成功标准：能解释total这一行为什么先乘后加。

休息5分钟：离开屏幕，回来学循环

5分钟

站起来，伸伸手，回来进入“循环”。

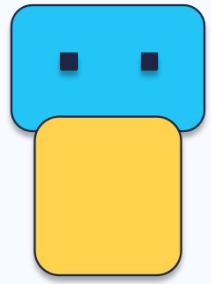
演示：老师运行一个倒计时程序。

提问：如果要打印10次Hello，你想写10行，还是让电脑重复？

休息回来就学：让电脑自动重复做事。

代码示例

```
for (int i = 5; i >= 1; i--) {
    printf("%d...\n", i);
}
printf("继续上课！\n");
```



编译器

控场：休息结束后，直接用“重复打印”引出循环。

循环：把重复任务交给电脑

6分钟

循环就是“只写一次规则，电脑重复执行”。

生活比喻：每天早上刷牙，动作相同但会重复。

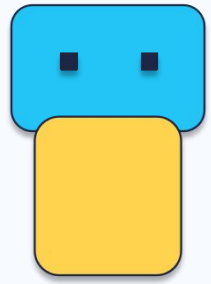
循环三件事：从哪里开始、什么时候停、每次怎么变化。

最常用：for循环。

适合：已知要重复多少次。

代码示例

```
for (int i = 1; i <= 5; i++) {
  printf("第%d次：Hello!\n", i);
}
```



编译器

for循环读法：从1开始；只要不超过5；每次加1。

for循环例子：从1加到10

6分钟

循环不只是重复打印，还能一点点累加结果。

sum像一个存钱罐。

每轮把i放进sum。

i从1走到10，sum越来越大。

最后sum就是总和。

快速猜一猜

如果 i <= 3，sum 最后是多少？

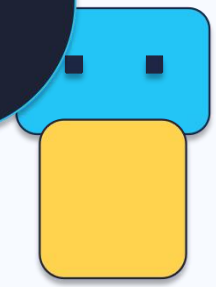
如果 i 每次 i += 2，会发生什么？

代码示例

```
int sum = 0;

for (int i = 1; i <= 10; i++) {
        sum = sum + i;
    }

printf("1到10的和是%d\n", sum);
```



编译器

核心动作：sum = sum + i 不是数学等式，是“用新结果更新变量”。

while循环：条件满足就继续

5分钟

不知道重复几次时，while更自然。

for：通常知道次数。

while：只要条件还成立，就继续。

break：立刻跳出循环。

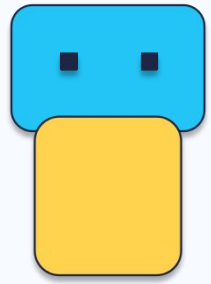
continue：跳过本轮后面的内容，进入下一轮。

代码示例

```
int energy = 3;

while (energy > 0) {
  printf("跑一圈! \n");
        energy--;
          }

printf("休息一下\n");
```



编译器

危险提醒：忘记让条件变化，可能出现停不下来的循环。

练习3：循环挑战

7分钟

二选一完成，速度快的同学做挑战题。

扫码打开在线C编译器

任务卡

基础A：打印1到10。

基础B：打印5、4、3、2、1倒计时。

挑战：计算1到100的偶数和。

提示：偶数可以用 i += 2，或者用 i % 2 == 0。

参考代码

```
// 倒计时
for (int i = 5; i >= 1; i--) {
    printf("%d\n", i);
}

// 偶数和提示
int sum = 0;
for (int i = 2; i <= 100; i += 2) {
    sum += i;
}
```

编译器

成功标准：能说出循环从哪里开始、什么时候停、每次怎么变。

函数：把代码打包成可重复使用的工具

4分钟

函数像一台机器：给它材料，它吐出结果。

为什么需要函数？同一段代码不用复制很多遍。

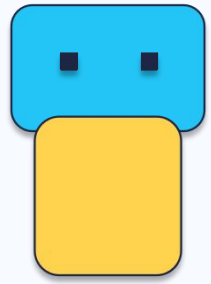
函数三件事：名字、参数、返回值。

printf其实也是函数，只是别人已经帮我们写好了。

今天只认识最基础的函数写法。

代码示例

```
返回类型 函数名(参数) {
    要做的事情;
    return 结果;
}
```



编译器

直觉：函数=可重复使用的工具。

函数例子：写一个加法工具

4分钟

把“加两个数”封装成一个函数。

add是函数名。

a和b是传进去的两个参数。

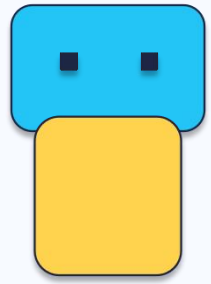
return把结果返回给调用它的地方。

练习：把add改成doubleScore，返回score * 2。

代码示例

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

int main() {
    int result = add(3, 5);
    printf("%d\n", result);
    return 0;
}
```



编译器

函数不用一次学透，先记住：写一次，可以用很多次。

收尾：今天你学会了哪些积木？

1分钟

能解释给别人听，才是真的开始理解。

数据类型：规定数据的存法。

运算符：完成计算和判断。

表达式：算出一个值。

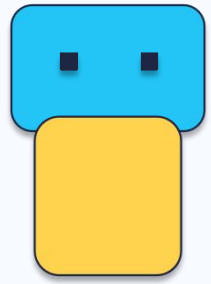
语句：让电脑执行一个动作。

循环：重复执行。

函数：把代码打包成工具。

出口测试

1. char和char[]有什么区别？
2. = 和 == 哪个是比较？
3. for循环的三件事是什么？
4. 函数为什么能减少重复代码？



编译器

结束语：报错不是失败，是电脑在告诉你哪里需要更精确。