程序设计
Programming

Lecture 2: C语言基本概念与元素

L01程序设计基本概念回顾

程序设计语言

抽象化

高级语言
(High-level language)

汇编语言
(Assembly language)

机器语言
(Machine language)

硬件系统
(Hardware)

编译器
(Compiler)

汇编器
(Assembler)

+

链接器
(Linker)

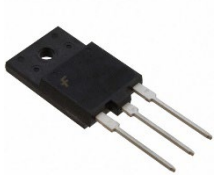
逻辑电路


```
mov ecx, ebx
mov esp, edx
mov edx, r9d
mov rax, rdx
```

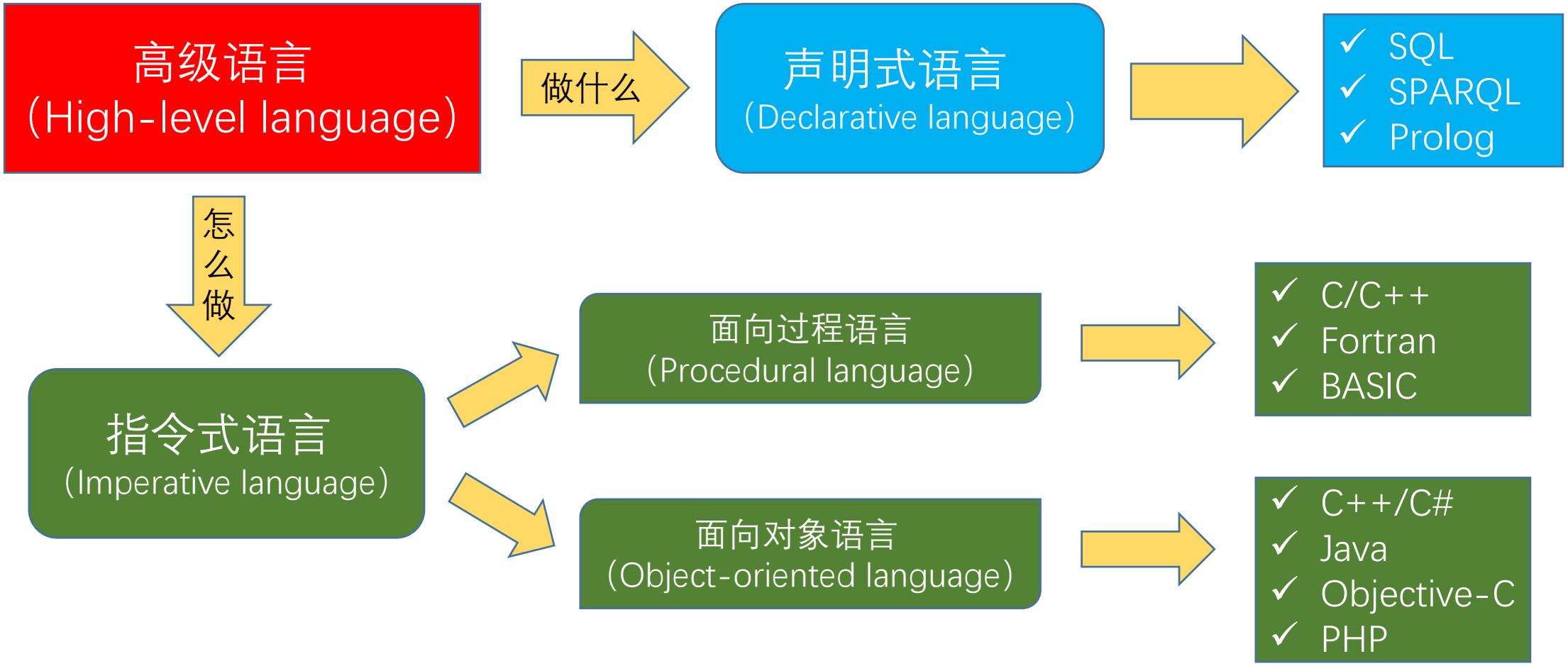
Assembler + Linker

```
100101011001
010011111011
111010101101
01010101010
```

```
0011000000000000 ; read n -> acc ;
1011000000001010 ; jump to Done if n < 0. ;
0101000000010000 ; add sum to the acc ;
0011000000010000 ; store the new sum ;
1001000000000000 ; go back & read in next number ;
0001000000010000 ; load the final sum ;
0100000000000000 ; output the final sum ;
0000000000000000 ; stop ;
0000000000000000 ; 2-byte location where sum is stored ;
0000000000000000  ;
0000000000000000  ;
```



高级语言分类


高级语言
(High-level language)
做什么
声明式语言
(Declarative language)
✓ SQL
✓ SPARQL
✓ Prolog
怎么做
指令式语言
(Imperative language)
面向过程语言
(Procedural language)
✓ C/C++
✓ Fortran
✓ BASIC
面向对象语言
(Object-oriented language)
✓ C++/C#
✓ Java
✓ Objective-C
✓ PHP

C语言

- C is a general-purpose, procedural computer programming language supporting structured programming, lexical scope, and recursion, while a static type system prevents unintended operations.

- 通用的：底层硬件、系统产品、数据分析、人工智能
- 面向过程：先做什么，后做什么（面向对象：你来做什么）
- 结构化编程：控制结构、区块、子程序
- 词法作用域：定义变量能够被使用的代码域
- 递归：将大问题分解为相同类型的子问题
- 静态类型系统：在程序运行前检查类型问题

C语言的发展历史

```
gcc main.c -o main.exe -std=c11
```

时间	C标准	事件
1972年	Birth	贝尔实验室Dennis Ritchie和Ken Thompson等人开发Unix系统时发明了C语言
1978年	K&R C	Brian Kernighan和Dennis Ritchie出版了第一本C语言教程《The C Programming Language》
1989年	ANSI C	美国国家标准学会和国际标准化组织（ISO）发布了C语言的标准（标准C）
1999年	C99	加入了内联函数、新的数据类型、可变长数组等新特性
2011年	C11	加入了通用宏、匿名结构、多线程和边界检查函数等新特性
2018年	C18	最新的C语言标准，对C11的缺陷进行了修订

一个最简单的C程序语言

- 预处理指令
  - ✓#include，#define，#ifdef，#ifndef等等
- 函数
  - ✓主函数，库函数，程序员自定义函数
- 语句
  - ✓条件语句，循环语句，返回语句等等

```
预处理指令

自定义函数（可以没有）

main ()
{
      语句
}
```

C语言代码中的基本元素

(标识符、运算符、表达式、语句、函数)

一、标识符（Identifier）

- 用来标识某个实体的符号，即给程序中的实体命名
  - ✓变量名，常量名，关键字（保留字），函数名，数据类型名等

- C语言标识符可以由字母、数字和下划线构成
  - ✓第一个字母必须是字母或者下划线，例如_name, b2c
  - ✓区分大小写，_name和_Name是不同的标识符

- 标识符命名应该“见名知意”
  - ✓长度（length），求和（sum），圆周率（pi）

一、标识符（Identifier）

- 用来标识某个实体的符号，即给程序中的实体命名
  - ✓**变量名，常量名，关键字（保留字），**函数名，数据类型名等

- C语言标识符可以由字母、数字和下划线构成
  - ✓第一个字母必须是字母或者下划线，例如_name, b2c
  - ✓区分大小写，_name和_Name是不同的标识符

- 标识符命名应该“见名知意”
  - ✓长度（length），求和（sum），圆周率（pi）

变量（Variable）

- 用于存储数据的存储单元
- 每一个变量必须有一个类型（或称数据类型），用来说明所存储数据的种类

变量（Variable）

- 用于存储数据的存储单元

- 每一个变量必须有一个类型（或称数据类型），用来说明所存储数据的种类
  - ✓整型（int，可以理解为整数），浮点型（float，可以理解为小数），字符型（char，单个字符）

变量（Variable）

- 用于存储数据的存储单元

- 每一个变量必须有一个类型（或称数据类型），用来说明所存储数据的种类
  - ✓整型（int，可以理解为整数），浮点型（float，可以理解为小数），字符型（char，单个字符）
  - ✓不同的类型变量占据的内存大小不同，所能存储数据的取值范围也不同

变量（Variable）

- 用于存储数据的存储单元

- 每一个变量必须有一个类型（或称数据类型），用来说明所存储数据的种类
  - ✓整型（int，可以理解为整数），浮点型（float，可以理解为小数），字符型（char，单个字符）
  - ✓不同的类型变量占据的内存大小不同，所能存储数据的取值范围也不同
  - ✓比如某些操作系统和编译器上，int型和float型都占据4个字节（但也有一些系统上，int型占据2个字节）

变量的声明

- 变量在使用前必须对其进行声明
  - ✓int i;
  - ✓float revenue;
  - ✓char ch;

变量的声明

- 变量在使用前必须对其进行声明
  - ✓int i;
  - ✓float revenue;
  - ✓char ch;

- 相同类型的变量的声明，可以放在一行
  - ✓int a, b, c, d;
  - ✓float revenue, debt;
  - ✓char ch1, ch2;

变量的声明

- 变量在使用前必须对其进行声明
  - ✓int i;
  - ✓float revenue;
  - ✓char ch;

- 相同类型的变量的声明，可以放在一行
  - ✓int a, b, c, d;
  - ✓float revenue, debt;
  - ✓char ch1, ch2;

- 变量命名规则与规范
  - ✓字母、数字和下划线的组合，不能以数字开头，不能使用关键字

变量的声明

- 变量在使用前必须对其进行声明
  - ✓int i;
  - ✓float revenue;
  - ✓char ch;

- 相同类型的变量的声明，可以放在一行
  - ✓int a, b, c, d;
  - ✓float revenue, debt;
  - ✓char ch1, ch2;

- 变量命名规则与规范
  - ✓字母、数字和下划线的组合，不能以数字开头，不能使用关键字
  - ✓尽量取有意义的名字：营收 r→revenue，我的体重 w→my_weight

变量的声明

- 变量在使用前必须对其进行声明
  - ✓int i;
  - ✓float revenue;
  - ✓char ch;

- 相同类型的变量的声明，可以放在一行
  - ✓int a, b, c, d;
  - ✓float revenue, debt;
  - ✓char ch1, ch2;

- 变量命名规则与规范
  - ✓字母、数字和下划线的组合，不能以数字开头，不能使用关键字
  - ✓尽量取有意义的名字：营收 r→revenue，我的体重 w→my_weight
  - ✓C语言区分大小写，即revenue和Revenue是不同的变量

变量的赋值

- 变量通过赋值存储相应数据类型的值
  - ✓i = 12;
  - ✓revenue = 50.00;
  - ✓ch = ‘s’; (ch = ‘1’;)

变量的赋值

- 变量通过赋值存储相应数据类型的值
  - ✓i = 12;
  - ✓revenue = 50.00;
  - ✓ch = ‘s’; (ch = ‘1’;)
- 赋值之后的变量可以进行运算，并将结果赋予其他变量
  - ✓float total_revenue = revenue * i;
  - ✓i = i + 1;

变量的赋值

- 变量通过赋值存储相应数据类型的值
  - ✓i = 12;
  - ✓revenue = 50.00;
  - ✓ch = ‘s’; (ch = ‘1’;)
- 赋值之后的变量可以进行运算，并将结果赋予其他变量
  - ✓float total_revenue = revenue * i;
  - ✓i = i + 1;
- 没有赋值的变量，在程序开始以后往往会被初始化为随机值（或零）！

变量的赋值

- 变量通过赋值存储相应数据类型的值
  - ✓i = 12;
  - ✓revenue = 50.00;
  - ✓ch = ‘s’; (ch = ‘1’;)
- 赋值之后的变量可以进行运算，并将结果赋予其他变量
  - ✓float total_revenue = revenue * i;
  - ✓i = i + 1;
- 没有赋值的变量，在程序开始以后往往会被初始化为随机值（或零）！

```c
#include<stdio.h>
int main(void)
{
    int m,n,i;
    double sum;
    scanf("%d%d",&m,&n);
    for(i=m;i<=n;i++)
        sum=sum+(i*i+1.0/i);

    printf("%.6lf",sum);

    return 0;
}
```

Status ● Wrong Answer

变量的赋值

- 变量通过赋值存储相应数据类型的值
  - ✓i = 12;
  - ✓revenue = 50.00;
  - ✓ch = ‘s’; (ch = ‘1’;)
- 赋值之后的变量可以进行运算，并将结果赋予其他变量
  - ✓float total_revenue = revenue * i;
  - ✓i = i + 1;
- 没有赋值的变量，在程序开始以后往往会被初始化为随机值（或零）！
  - ✓可以在声明的同时就给变量赋值，即变量初始化：double sum = 0;

```c
#include<stdio.h>
int main(void)
{
    int m,n,i;
    double sum;
    scanf("%d%d",&m,&n);
    for(i=m;i<=n;i++)
        sum=sum+(i*i+1.0/i);

    printf("%.6lf",sum);

    return 0;
}
```

Status ● Wrong Answer

变量的赋值

- 变量通过赋值存储相应数据类型的值
  - ✓i = 12;
  - ✓revenue = 50.00;
  - ✓ch = ‘s’; (ch = ‘1’;)
- 赋值之后的变量可以进行运算，并将结果赋予其他变量
  - ✓float total_revenue = revenue * i;
  - ✓i = i + 1;
- 没有赋值的变量，在程序开始以后往往会被初始化为随机值（或零）！
  - ✓可以在声明的同时就给变量赋值，即变量初始化: double sum = 0;
  - ✓int weight = 75, height = 183;
  - ✓float weight, height = 1.83;（weight没有赋值！！）

```c
#include<stdio.h>
int main(void)
{
    int m,n,i;
    double sum;
    scanf("%d%d",&m,&n);
    for(i=m;i<=n;i++)
        sum=sum+(i*i+1.0/i);

    printf("%.6lf",sum);

    return 0;
}
```

Status ● Wrong Answer

变量代码示例

variable.c

C > L2 > C variable.c > main()

```c
/*变量演示*/
#include <stdio.h>

int main()
{
    int number_of_months = 12;
    float revenue = 50.00;
    float total_revenue = number_of_months * revenue;
    number_of_months = number_of_months + 1;

    //int a;
    //printf("%d\n\n", a);
    printf("%f, %d\n\n", total_revenue, number_of_months);

    return 0;
}
```

常量（constant）

- 在程序执行过程中固定不变的量
  - ✓不同数据类型的常量：25，123.4，’a’，”hello world”

常量（constant）

- 在程序执行过程中固定不变的量
  - ✓不同数据类型的常量：25，123.4，’a’，”hello world”

- 给常量命名，方便理解和重复使用
  - ✓命名规则同变量命名规则
  - ✓方式一：预处理阶段，使用宏定义，#define PI 3.14159（PI是3.14159的别名）
  - ✓方式二：语句中，const float pi = 3.14159（pi是一个值固定不变的变量）

常量（constant）

- 在程序执行过程中固定不变的量
  - ✓不同数据类型的常量：25，123.4，’a’，”hello world”

- 给常量命名，方便理解和重复使用
  - ✓命名规则同变量命名规则
  - ✓方式一：预处理阶段，使用宏定义，#define PI 3.14159（PI是3.14159的别名）
  - ✓方式二：语句中，const float pi = 3.14159（pi是一个值固定不变的变量）
  - ✓能否能再次赋值？

const定义的常量不可以重新赋值！
#define定义的常量可以通过#undef和#define重新赋值

常量（constant）

- 在程序执行过程中固定不变的量
  - ✓不同数据类型的常量：25，123.4，’a’，”hello world”
- 给常量命名，方便理解和重复使用
  - ✓命名规则同变量命名规则
  - ✓方式一：预处理阶段，使用宏定义，#define PI 3.14159（PI是3.14159的别名）
  - ✓方式二：语句中，const float pi = 3.14159（pi是一个值固定不变的变量）
  - ✓能否能再次赋值?
- 使用常量（和使用变量类似）
  - ✓float perimeter = 2 * pi * r

常量代码示例

constant.c

C > L2 > C constant.c > main()

```c
#include <stdio.h>

int main()
{
    const float pi = 3.14159;
    int r = 2;
    //pi = 3.14;
    float perimeter = 2 * pi * r;
    printf("%f\n\n", perimeter);

    return 0;
}
```

关键字（保留字）

- C语言专门规定的，具有特殊意义和专门用途的标识符
  - ✓int, float
  - ✓if, else, for
  - ✓typedef

- ANSI C定义的32个关键字

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

二、运算符（Operator）

- 对各种数据类型进行运算

- 运算符按功能分类
  - ✓算术运算符：+, -, *, /, %, ++(自增), --(自减)
  - ✓关系运算符：>, >=, <, <=, ==, !=
  - ✓逻辑运算符：&&(与), ||(或), !(非), ~, &, |, ^
  - ✓赋值运算符：=
  - ✓移位运算符：<<, >>

- 运算符按操作数分类
  - ✓一元（目）运算符: +, -, ++, -- 等
  - ✓二元（目）运算符: +, -, *, /, %, >, >=, <, <=, ==, != 等
  - ✓三元（目）运算符: 布尔表达式？表达式1 : 表达式2 （条件运算符）

二、运算符（Operator）

- 对各种数据类型进行运算

- 运算符按功能分类
  - ✓算术运算符：+, -, *, /, %, ++(自增), --(自减)
  - ✓关系运算符：>, >=, <, <=, ==, !=
  - ✓逻辑运算符：&&(与), ||(或), !(非), ~, &, |, ^
  - ✓赋值运算符：=
  - ✓移位运算符：<<, >>

- 运算符按操作数分类
  - ✓一元（目）运算符: +, -, ++, -- 等
  - ✓二元（目）运算符: +, -, *, /, %, >, >=, <, <=, ==, != 等
  - ✓三元（目）运算符: 布尔表达式？表达式1 : 表达式2 （条件运算符）

二、运算符（Operator）

- 对各种数据类型进行运算

- 运算符按功能分类
  - ✓算术运算符：+, -, *, /, %, ++(自增), --(自减)
  - ✓关系运算符：>, >=, <, <=, ==, !=
  - ✓逻辑运算符：&&(与), ||(或), !(非), ~, &, |, ^
  - ✓赋值运算符：=
  - ✓移位运算符：<<, >>

掌握红色部分

- 运算符按操作数分类
  - ✓一元（目）运算符: +, -, ++, -- 等
  - ✓二元（目）运算符: +, -, *, /, %, >, >=, <, <=, ==, != 等
  - ✓三元（目）运算符: 布尔表达式？表达式1 : 表达式2 （条件运算符）

二、运算符（Operator）

- 对各种数据类型进行运算

- 运算符按功能分类
  - ✓算术运算符：+, -, *, /, %, ++(自增), --(自减)
  - ✓关系运算符：>, >=, <, <=, ==, !=
  - ✓逻辑运算符：&&(与), ||(或), !(非), ~, &, |, ^
  - ✓赋值运算符：=
  - ✓移位运算符：<<, >>

掌握红色部分

注意不要混淆赋值运算(=)和关系运算(==)

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序
- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序
  - ✓左结合，从左向右结合：除了右结合运算符，都是左结合

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序
  - ✓左结合，从左向右结合：除了右结合运算符，都是左结合
  - ✓右结合，从右向左结合：一元算术运算符，赋值运算符和条件运算符等

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序
  - ✓左结合，从左向右结合：除了右结合运算符，都是左结合
  - ✓右结合，从右向左结合：一元算术运算符，赋值运算符和条件运算符等
  - ✓i + j – k 等价于 (i + j) – k

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序
  - ✓左结合，从左向右结合：除了右结合运算符，都是左结合
  - ✓右结合，从右向左结合：一元算术运算符，赋值运算符和条件运算符等
  - ✓i + j – k 等价于 (i + j) – k
  - ✓- + i 等价于 – (+ i)

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序
  - ✓左结合，从左向右结合：除了右结合运算符，都是左结合
  - ✓右结合，从右向左结合：一元算术运算符，赋值运算符和条件运算符等
  - ✓i + j – k 等价于 (i + j) – k
  - ✓- + i 等价于 – (+ i)
  - ✓-++i 等价于？

运算符的优先级和结合性

- 定义了执行多个运算的先后顺序

- 首先使用优先级定义运算顺序
  - ✓*和/优先级高于+和-，例如：i + j * k 等价于 i + (j * k)
  - ✓一元运算符优先级高于二元和三元运算符
  - ✓例如：-i * -j 等价于 (-i) * (-j)

- 当优先级相同的运算符在一起时，根据结合性定义顺序
  - ✓左结合，从左向右结合：除了右结合运算符，都是左结合
  - ✓右结合，从右向左结合：一元算术运算符，赋值运算符和条件运算符等
  - ✓i + j – k 等价于 (i + j) – k
  - ✓- + i 等价于 – (+ i)
  - ✓-++i 等价于？
  - ✓a = b = c 等价于？

C语言运算符优先级和结合性表

附录C　C语言运算符优先级和结合性

优先级	运算符	含义	运算类型	结合性
1	( ) [ ] -> .	圆括号 下标运算符 指向结构体成员运算符 结构体成员运算符	单目	自左向右
2	! ~ ++ -- (类型关键字) + - * & sizeof	逻辑非运算符 按位取反运算符 自增、自尖运算符 强制类型转换 正、负号运算符 指针运算符 地址运算符 长度运算符	单目	自右向左
3	* / %	乘、除、求余运算符	双目	自左向右
4	+ -	加、减运算符	双目	自左向右
5	<< >>	左移运算符 右移运算符	双目	自左向右
6	< <= > >=	小于、小于等于、大于、大于等于	关系	自左向右
7	== ! =	等于、不等于	关系	自左向右
8	&	按位与运算符	位运算	自左向右
9	^	按位异或运算符	位运算	自左向右
10	|	按位或运算符	位运算	自左向右
11	&&	逻辑与运算符	位运算	自左向右
12	||	逻辑或运算符	位运算	自左向右
13	? :	条件运算符	三目	自右向左
14	= += -= *= /= %= << = >>= &= ^= |=	赋值运算符	双目	自右向左
15	,	逗号运算	顺序	自左向右

运算符代码示例

C operator.c ×

C > L2 > C operator.c > main()

```c
#include <stdio.h>

int main()
{
    int a = 2;
    a++;
    printf("%d\n", a);
    int b = ++a * 3;
    printf("%d, %d\n", a, b);
    printf("%d\n", a++);
    printf("%d\n\n", ++a);

    return 0;
}
```

三、表达式(expression)和语句(statement)

- 运算符和运算对象（变量、常量、函数等）构成了表达式
  - ✓weight = 75.5
  - ✓2 * pi * r
  - ✓运算符的优先级确定了表达式的执行顺序

- 表达式构成C语言的语句
  - ✓任何表达式添加分号（;）以后就可以变成语句
  - ✓通常来讲，语句由一个或一组执行完整功能的表达式构成

四、函数（以格式化输入与输出为例）

- 与终端进行交互，使用C标准库中的函数
  - ✓输出函数printf()
  - ✓输入函数scanf()

四、函数（以格式化输入与输出为例）

- 与终端进行交互，使用C标准库中的函数
  - ✓输出函数printf()
  - ✓输入函数scanf()

- printf(“The dog is %d years old.\n”, age)
  - ✓用特定格式向终端打印输出

四、函数（以格式化输入与输出为例）

- 与终端进行交互，使用C标准库中的函数
  - ✓输出函数printf()
  - ✓输入函数scanf()

- printf(“The dog is %d years old.\n”, age)
  - ✓用特定格式向终端打印输出
  - ✓“The dog is %d years old”是格式串（format string），定义打印输出的样式

四、函数（以格式化输入与输出为例）

- 与终端进行交互，使用C标准库中的函数
  - ✓输出函数printf()
  - ✓输入函数scanf()

- printf(“The dog is %d years old.\n”, age)
  - ✓用特定格式向终端打印输出
  - ✓“The dog is %d years old”是格式串（format string），定义打印输出的样式
  - ✓The dog is 和 years old 是普通字符串；\n是转义字符；%d是转换说明，指定了将数值从二进制转换成打印字符的方法（十进制）

四、函数（以格式化输入与输出为例）

- 与终端进行交互，使用C标准库中的函数
  - ✓输出函数printf()
  - ✓输入函数scanf()

- printf(“The dog is %d years old.\n”, age)
  - ✓用特定格式向终端打印输出
  - ✓“The dog is %d years old”是格式串（format string），定义打印输出的样式
  - ✓The dog is 和 years old 是普通字符串；\n是转义字符；%d是转换说明，指定了将数值从二进制转换成打印字符的方法（十进制）
  - ✓age：要打印输出的数值。可以是变量，常量，或者函数

四、函数（以格式化输入与输出为例）

- 与终端进行交互，使用C标准库中的函数
  - ✓输出函数printf()
  - ✓输入函数scanf()

- printf(“The dog is %d years old.\n”, age)
  - ✓用特定格式向终端打印输出
  - ✓“The dog is %d years old”是格式串（format string），定义打印输出的样式
  - ✓The dog is 和 years old 是普通字符串；\n是转义字符；%d是转换说明，指定了将数值从二进制转换成打印字符的方法（十进制）
  - ✓age：要打印输出的数值。可以是变量，常量，或者函数
  - ✓一次调用printf可以打印的值的个数没有限制，转换说明和打印数值按顺序一一对应

scanf()

- scanf(“%d”, &age)
  - ✓从终端根据特定格式读取输入

scanf()

- scanf(“%d”, &age)
  - ✓从终端根据特定格式读取输入
  - ✓“%d”是格式串，把输入字符与转换说明匹配（跳过输入的空白字符，即空格，回车和Tab等）

scanf()

- scanf(“%d”, &age)
  - ✓从终端根据特定格式读取输入
  - ✓“%d”是格式串，把输入字符与转换说明匹配（跳过输入的空白字符，即空格，回车和Tab等）
  - ✓一次调用scanf()可以匹配的值的个数没有限制，转换说明和匹配字符按顺序一一对应

scanf()

- scanf(“%d”, &age)
  - ✓从终端根据特定格式读取输入
  - ✓“%d”是格式串，把输入字符与转换说明匹配（跳过输入的空白字符，即空格，回车和Tab等）
  - ✓一次调用scanf()可以匹配的值的个数没有限制，转换说明和匹配字符按顺序一一对应
  - ✓scanf()格式串往往只包含转换说明，如果里面包含普通字符，必须“原样”输入

scanf()

- scanf(“%d”, &age)
  - ✓从终端根据特定格式读取输入
  - ✓“%d”是格式串，把输入字符与转换说明匹配（跳过输入的空白字符，即空格，回车和Tab等）
  - ✓一次调用scanf()可以匹配的值的个数没有限制，转换说明和匹配字符按顺序一一对应
  - ✓scanf()格式串往往只包含转换说明，如果里面包含普通字符，必须“原样”输入
  - ✓&age：age用来存储匹配到的字符，&age是age在内存中的地址

scanf()

- scanf(“%d”, &age)
  - ✓从终端根据特定格式读取输入
  - ✓“%d”是格式串，把输入字符与转换说明匹配（跳过输入的空白字符，即空格，回车和Tab等）
  - ✓一次调用scanf()可以匹配的值的个数没有限制，转换说明和匹配字符按顺序一一对应
  - ✓scanf()格式串往往只包含转换说明，如果里面包含普通字符，必须“原样”输入
  - ✓&age：age用来存储匹配到的字符，&age是age在内存中的地址

- 常用的转换说明
  - ✓有符号十进制整数 %d，浮点数 %f, %lf，字符 %c，字符串 %s
  - ✓无符号十进制整数 %u，长整型 %ld, %lu

转换说明符表

C标准转换说明,表一

	C语言prinf转换说明及其打印结果
转换说明	输出
%a	浮点数,十六进制和p记数法(C99/C11)
%A	浮点数,十六进制和p记数法(C99/C11)
%c	单个字符
%d	有符号十进制
%e	浮点数, e记数法
%E	浮点数, e记数法
%f	浮点数, 十进制记数法
%g	根据值的不同,自动选择%f或%e, %e格式用于指数小于-4或大于或等于精度时
%G	根据值的不同,自动选择%f或%e, %e格式用于指数小于-4或大于或等于精度时
%i	有符号十进制(%d相同)
%o	无符号八进制
%p	指针
%s	字符串
%u	无符号十进制整数
%x	无符号十六进制整数
%X	无符号十六进制整数
%%	打印一个百分号

转换说明符的完整格式

- %m.pX，m和p都是整数常量，X是字母（转换格式）
  - ✓用法：%m.pX 或 %mX 或 %.pX
  - ✓m：最小字段宽度，即要显示的最小字符数，不足的用空格在左侧补齐
  - ✓p：精度，具体含义和X有关，比如%.pf指小数点后保留位数，%.pd指显示数字最少个数，不足的用0在左侧补齐

转换说明符的完整格式

- %m.pX，m和p都是整数常量，X是字母（转换格式）
  - ✓用法：%m.pX 或 %mX 或 %.pX
  - ✓m：最小字段宽度，即要显示的最小字符数，不足的用空格在左侧补齐
  - ✓p：精度，具体含义和X有关，比如%.pf指小数点后保留位数，%.pd指显示数字最少个数，不足的用0在左侧补齐
  - ✓float a = 12.3
  - ✓%f ?
  - ✓%7.2f ?
  - ✓int a = 5
  - ✓%7.2d ?

格式化输入输出代码示例

inputOutput.c

C > L2 > C inputOutput.c > main()

```c
#include <stdio.h>

int main()
{
    int age;
    float weight;
    scanf("%d%f", &age, &weight);
    printf("Xiaoming is %d years old and %f kg.\n\n", age, weight);
    printf("Xiaoming is %d years old and %.0f kg.\n\n", age, weight);  //小数点后保留0位

    return 0;
}
```

预：数据类型（L03）

- 字符型 char，1字节，取值范围 -128 ~ 127
- 短整型 short，2字节，取值范围 -32768~32767
- 整型 int，4字节，取值范围 -2147483648 ~ 2147483647
- 长整型 long，8字节
- 单精度浮点型 float，4字节，取值范围 -3.4E38 ~ 3.4E38，大约精确到小数点后7位
- 双精度浮点型 double，8字节，取值范围 -1.7E308 ~ 1.7E308，大约精确到小数点后15位

https://www.tutorialspoint.com/c_standard_library/limits_h.htm

有符号和无符号

- 有符号数
  - ✓在类型（整型和字符型）前加signed，signed char = -10
- 无符号数
  - ✓在类型（整型和字符型）前加unsigned，unsigned int = 20
- 通常默认为有符号数
- 无符号数取值范围
  - ✓char, 0 ~ 255
  - ✓short, 0~ 65535
  - ✓int, 0 ~ 4294967295

类型转换

- 数据类型之间可以互相转换
  - ✓显式转换：int a = (int) 12.3
  - ✓隐式转换：float a = 12； int a = 12.3； int a = ‘A’

- 从存储范围大的数据类型转换到存储类型小的数据类型，会发生精度丢失!

小结

- C语言的基本概念
  - ✓完整的C程序至少包括：预处理，（主）函数和语句
  - ✓C程序中的基本元素：标识符，运算符，表达式，输入输出，数据类型

- C语言的概念很简单，使用灵活性极高，多从网上找答案，多实践

L03：数据类型与控制结构