# LECTURE 9 期末复习

工学院18-19学年秋季学期计算概论（邓习峰班）课后辅导

讲师：陈婉雯

日期：2018/12/29

# 表达式和语句

- 表达式是一种有值的语法结构，表达式是由一系列操作符（operators）和操作数（operands）组成的。
- 语句是程序运行时执行的命令。C语言标准规定语句以;结尾，但是对于复合语句，它用大括号{}将多条语句包裹起来，强制编译器将其当作一条语句处理，结尾不需要;。

# 运算符优先级

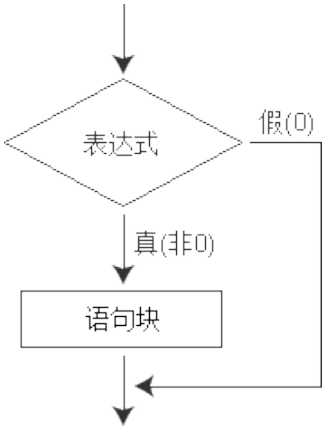
- ！ > 算术运算符 > 关系运算符 > && > || >赋值运算符
- https://blog.csdn.net/sunshihua12829/article/details/47912123

# C语言的三种程序结构

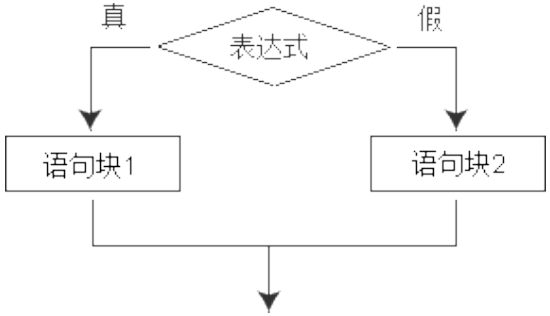
- 顺序结构
- 分支结构
  - if
  - switch
- 循环结构
  - for
  - while
  - do...while

# 分支结构-if

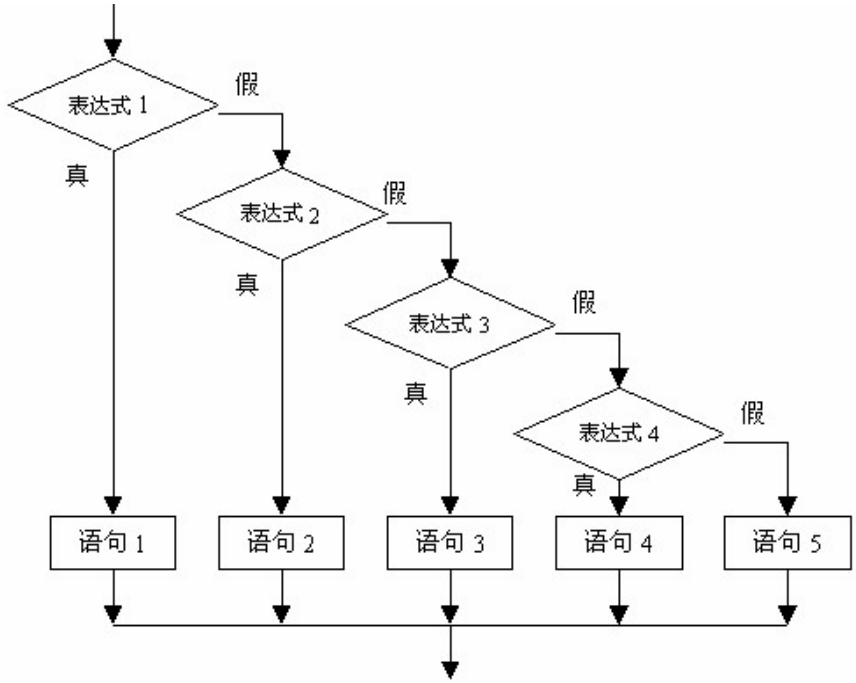
- if




if - else




if - else if - else



# if后的关键字

- if关键字之后均为表达式。
  - 该表达式通常是逻辑表达式或关系表达式，但也可以是其它表达式，如赋值表达式等，甚至也可以是一个变量。
- 只要表达式的值为非0，即为“真”。
  - if(a=5)...;表达式的值永远为非0，所以其后的语句总是要执行的
- 想在满足条件时执行一组(多个)语句，则必须把这一组语句用{}括起来组成一个复合语句。

```
if(a=b)
    printf("%d",a);
else
    printf("a=0");
```

```
if(a>b){a++;
    b++;
}else{
    a=0;
    b=10;
}
```

# if的嵌套

- else总是与它前面最近的if配对

```
#include <stdio.h>
int main(void){
    int a,b;
    printf("please input A,B:    ");
    scanf("%d%d",&a,&b);
    if(a!=b)
    if(a>b)  printf("A>B\n");
    else     printf("A<B\n");
    else     printf("A=B\n");
    return 0;
}
```

# switch

- 计算表达式的值 并逐个与其后的常量表达式值相比较
- 当表达式的值与某个常量表达式的值相等时， 即执行其后的语句，然后不再进行判断，继续执行后面所有case后的语句。如表达式的值与所有case后的常量表达式均不相同时，则执行default后的语句。
- 每一case语句之后增加break 语句， 使每一次执行之后均可跳出switch语句，从而避免输出不应有的结果。

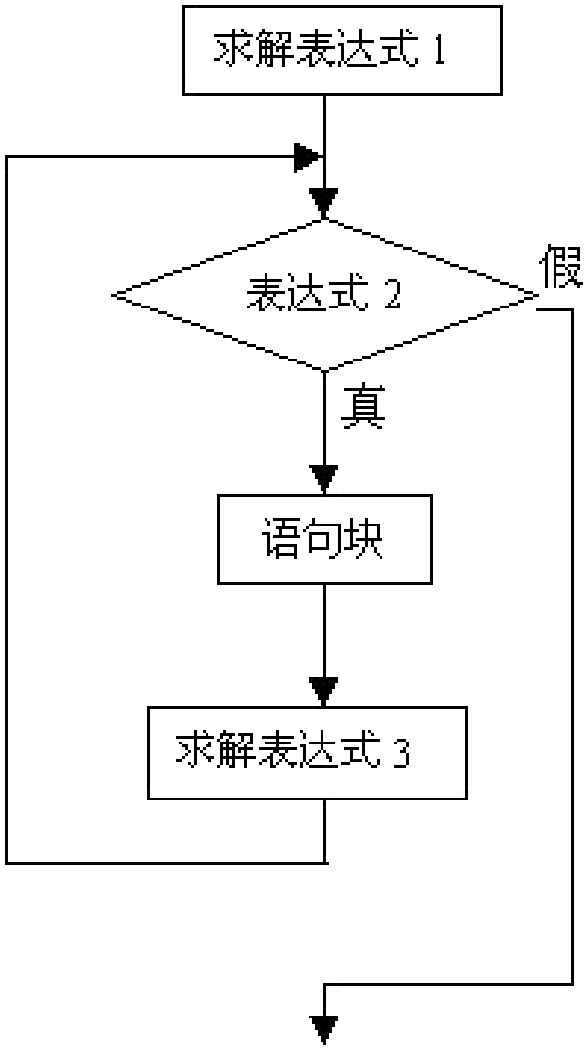
```
switch(表达式){
    case 常量表达式1: 语句1;
    case 常量表达式2: 语句2;
    ...
    case 常量表达式n: 语句n;
    default: 语句n+1;
}
```

# switch注意事项

- 在case后的各常量表达式的值不能相同，否则会出现错误。
- 在case后，允许有多个语句，可以不用{}括起来。
- 各case和default子句的先后顺序可以变动，而不会影响程序执行结果（在每个case都包含break的情况下）。
- 不是每一个 case 都需要包含 break。如果 case 语句不包含 break，控制流将会继续后续的 case，直到遇到 break 为止。
- 一个 switch 语句可以有一个可选的 default case，出现在 switch 的结尾。default case 可用于在上面所有 case 都不为真时执行一个任务。default case 中的 break 语句不是必需的。default子句可以省略不用。

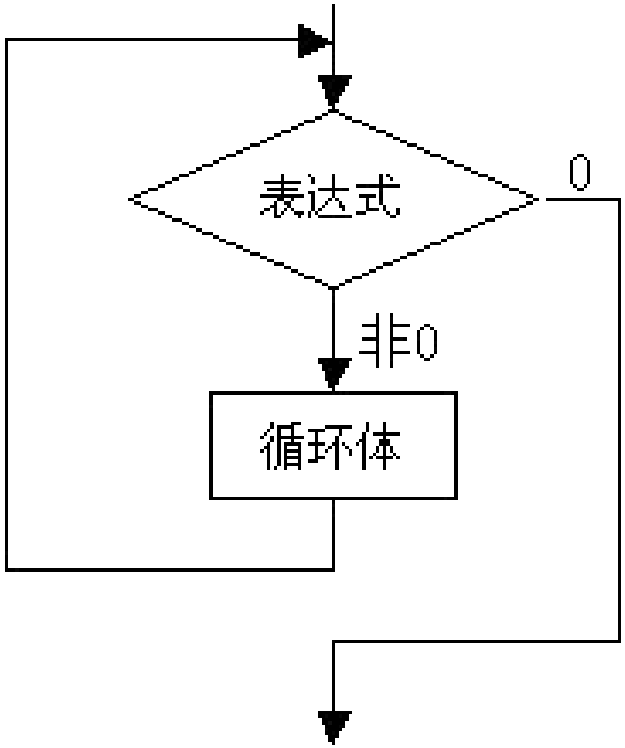
# for循环

- for(表达式1; 表达式2; 表达式3)
  - for(循环变量赋初值; 循环条件; 循环变量增量)
- for循环中的“表达式1（循环变量赋初值）”、“表达式2(循环条件)”和“表达式3（循环变量增量）”都是选择项，即可以缺省，但分号(;)不能缺省。
  - 省略了“表达式1（循环变量赋初值）”，表示不对循环控制变量赋初值
  - 省略了“表达式2(循环条件)”，则不做其它处理时便成为死循环
  - 省略了“表达式3(循环变量增量)”，则不对循环控制变量进行操作，这时可在语句体中加入修改循环控制变量的语句。

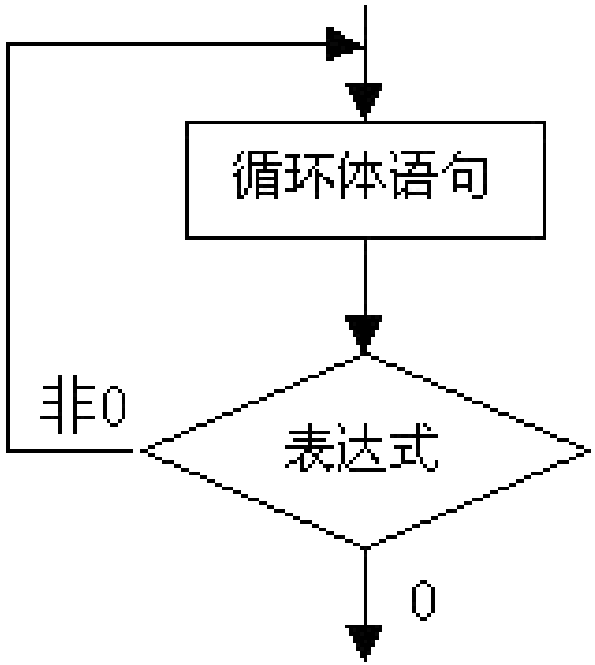


# while, do - while

- while(表达式)
  语句




- do
  语句
  while(表达式);



# 循环注意事项

- 循环终止表达式为0还是1（和if类似）
  - while(a=1)
  - while(a=b)
- 循环体如包括有一个以上的语句，则必须用{}括起来，组成复合语句。
- 循环嵌套

# 特殊结构语句

- break
- continue
- goto

# break

- 通常用在循环语句和switch语句中
- 当break语句用于do-while、for、while循环语句中时，可使程序终止循环而执行循环后面的语句，通常break语句总是与if语句联在一起，即满足条件时便跳出循环
- 当break用于switch中时，可使程序跳出switch而执行switch以后的语句
- break语句对if-else的条件语句不起作用
- 在多层循环中，一个break语句只向外跳一层

# continue

- continue语句的作用是跳过循环体中剩余的语句而强行执行下一次循环。continue语句只用在for、while、do-while等循环体中，常与if条件语句一起使用，用来加速循环。

- break与continue的对比：break 用来结束所有循环，循环语句不再有执行的机会；continue 用来结束本次循环，直接跳到下一次循环，如果循环条件成立，还会继续循环。

# goto

- C 语言中的 goto 语句允许把控制无条件转移到同一函数内的被标记的语句。
- goto 语句标号;

```c
#include <stdio.h>
int main(void){
    int i,sum=0;
    i=1;
    loop: if(i<=100){
        sum=sum+i;
        i++;
        goto loop;
    }
    printf("%d\n",sum);
    return 0;
}
```

# 函数

- 函数是一组一起执行一个任务的语句。
- 主函数main()
  - 程序开始执行的地方
- 函数定义
  - 函数头：返回类型、函数名称、参数
  - 函数主体
- 函数声明
  - 告诉编译器函数名称和调用方式
  - 可以不写参数名称（但一定要写类型）
- 函数调用
- 函数参数
  - 形式参数：和函数内的其他局部变量一样，进入函数时被创建，退出函数时被销毁
  - 用数组、指针时才能修改实际参数

```
return_type function_name( parameter list )
{
   body of the function
}
```

```
return_type function_name( parameter list );
```

# 考试易错点

- 循环、选择结构判断终止条件是否非0
  - 赋值运算符(=)和逻辑判断(==)
- switch的case是否执行
  - 注意有无break，如果没有的话会把之后的case都执行
- 函数形式参数和实际参数的值的传递
  - 一般来说，当函数参数为指针/数组时才能改变实际参数的值
- 运算符优先级
- 整型除法、求余

# 数据类型

- 类型
  - 字符
    - ASCII码、大小写转换、和整型的关系
    - 单引号
  - 整型
    - 进制转换与格式（八进制以0开头，十六进制以0x开头）
    - 输出控制符
  - 浮点型
    - 十进制表达、指数表达
- 存储单位
  - 位（0/1）字节（8位）
  - 整型（4字节）字符型（1字节）
- 强制数据类型转换：（类型说明符）（表达式）

# 常量和变量

- 常量：不能改的量
  - const关键字
  - 符号常量#define
  - 注意字符、字符串常量区别（单引号、双引号）
- 变量：可以改变的量
  - 声明、初始化

# 数组和字符串

- 创建数组
- 调用数组
- 高维数组
  - 只能省略最高维

# 结构体

- 定义结构体变量
- 创建结构体变量
- 初始化结构
- 访问结构成员
  - 注意通过结构体指针访问的方式
  - struct stuff Alex; Alex.number;
  - struct stuff *Alex; Alex->number; (*Alex).number

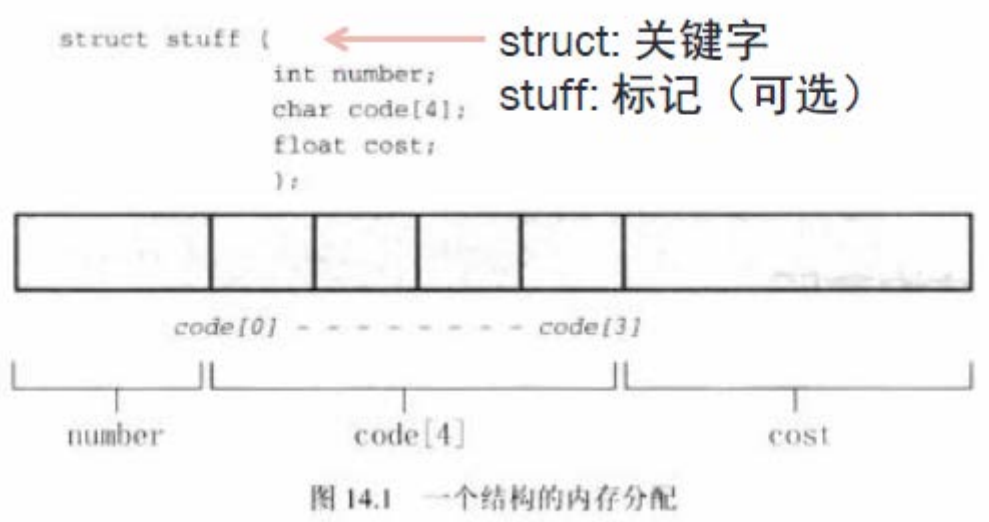


图 14.1 一个结构的内存分配

# 指针和内存分配

- 定义指针变量
- 通过malloc、free动态分配内存给指针
  - 记得malloc要强制类型转换
- 常量指针和指针常量
  - 常量指针 int const* p; const int * p
  - 指向的对象不能够通过这个指针来修改，既可以指向常量也可指向变量，指向的地址也可以修改
  - 不能把常量值赋给普通指针
  - 指针常量 int* const p;
  - 指向的对象可以通过这个指针修改，但不能指向不同的地址
- 函数指针
  - 用作函数参数，声明格式 int (*f)(int); 初始化f=func; 调用a=f(x)等价于a=func(x);

# 链表

- 基本操作：
- 创建
- 添加成员
- 删除成员
- 遍历链表
- 删除链表

# 输入输出

- 普通输入输出
- 文件输入输出
- 转义字符
- 输入时要取地址
- 注意格式控制
  - 特别是整型、浮点型里面控制输入输出格式（占多少位、左对齐和右对齐、几位数字、几位小数等）

表 4.3 转换说明符及作为结果的打印输出

| 转换说明 | 输出 |
|---|---|
| %a | 浮点数、十六进制数字和 p-记数法（C99） |
| %A | 浮点数、十六进制数字和 P-记数法（C99） |
| %c | 一个字符 |
| %d | 有符号十进制整数 |
| %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 | 根据数值不同自动选择%f 或%e。%e 格式在指数小于-4 或者大于等于精度时使用 |
| %G | 根据数值不同自动选择%f 或%E。%E 格式在指数小于-4 或者大于等于精度时使用 |
| %i | 有符号十进制整数（与%d 相同） |
| %o | 无符号八进制整数 |
| %p | 指针 |
| %s | 字符串 |
| %u | 无符号十进制整数 |
| %x | 使用十六进制数字 0f 的无符号十六进制整数 |
| %X | 使用十六进制数字 0F 的无符号十六进制整数 |
| %% | 打印一个百分号 |

表 4.4 printf()的修饰符

| 修饰符 | 含义 |
| --- | --- |
| 标记 | 表 4.5 描述了 5 种标记（-、+、空格、#和 0），可以不使用标记或使用多个标记<br>示例："%-10d" |
| 数字 | 最小字段宽度<br>如果该字段不能容纳待打印的数字或字符串，系统会使用更宽的字段<br>示例："%4d" |
| .数字 | 精度<br>对于%e、%E 和%f 转换，表示小数点右边数字的位数<br>对于%g 和%G 转换，表示有效数字最大位数<br>对于%s 转换，表示待打印字符的最大数量<br>对于整型转换，表示待打印数字的最小位数<br>如有必要，使用前导 0 来达到这个位数<br>只使用.表示其后跟随一个 0，所以%.f 和%.0f 相同<br>示例："%5.2f"打印一个浮点数，字段宽度为 5 字符，其中小数点后有两位数字 |

表 4.5 printf()中的标记

| 标记 | 含义 |
| --- | --- |
| - | 待打印项左对齐。即，从字段的左侧开始打印该项项<br>示例："%-20s" |
| + | 有符号值若为正，则在值前面显示加号；若为负，则在值前面显示减号<br>示例："%+6.2f" |
| 空格 | 有符号值若为正，则在值前面显示前导空格（不显示任何符号）；若为负，则在值前面显示减号<br>+标记覆盖一个空格<br>示例："%6.2f" |
| # | 把结果转换为另一种形式。如果是%o 格式，则以 0 开始；如果是%x 或%X 格式，则以 0x 或 0X 开始；对于所有的浮点格式，#保证了即使后面没有任何数字，也打印一个小数点字符。对于%g 和%G 格式，#防止结果后面的 0 被删除<br>示例："%#o"、"%#8.0f"、"%+#10.3e" |
| 0 | 对于数值格式，用前导 0 代替空格填充字段宽度。对于整数格式，如果出现-标记或指定精度，则忽略该标记 |

# 计算机发展史和常识

- 什么是冯诺依曼体系
- 原码、反码、补码
- 进制转换