第五章 控件流

一、三大控制结构简介

二、语句和块

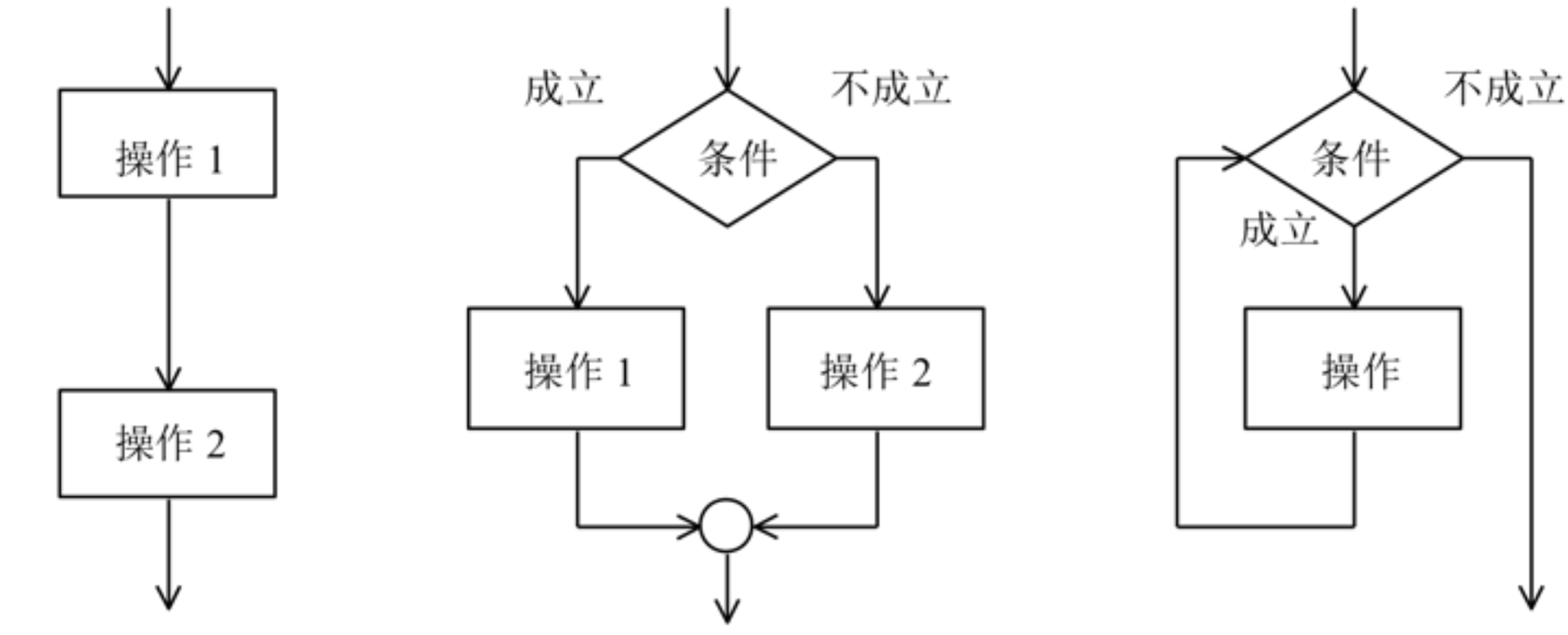
三、选择结构

四、循环结构

五、break/continue

一、C语言三大控制结构

控制流指定了语句的执行顺序，在C语言中主要有三大控制结构：顺序结构、选择结构和循环结构。



(a) 顺序执行　(b) 选择执行　(c) 循环执行

程序控制流程的三种基本模式

三大控制结构：

- **顺序结构：**按照语句编写的顺序逐句执行。
- **选择结构：**

(1) if选择结构——条件语句为真时，执行动作；条件语句为假，跳过不执行。

(2) if/else选择结构——条件语句为真时，执行if分支的动作；条件语句为假，执行else分支。

(3) switch选择结构——根据条件表达式的值，执行多个动作当中的某一个动作。

- **循环结构：**

(1) while循环结构

(2) do/while循环结构

(3) for循环结构

二、语句和块

一个表达式如 x = 0 或 i++ 或 printf（...）等在其后跟随一个分号则构成语句，语句是程序的基本单位，是由分号结束的一段字符，即分号是语句结束的标志。

复合语句:复合语句是把多个语句用花括号“{}”括起来组成 一个整体效果的执行语句(“}”后面不用加“;”)。其一般形式为:

```
{
    单条语句 …
}
```

复合语句内的单条语句都必须以分号“;”结尾,在括号“}”外面不需要有分号。例如:

```
{
    int x = 0;
    x++;
    printf(“x = %d\n”, x);
}
```

三、选择结构

3.1 if语句

用 if 语句可以构成分支语句结构。它根据给定的条件进行判断,以决定是否执行某个分支程序段。

C语言的 if 分支语句有三种基本形式:

- 第一种形式：if语句，如：if(表达式) {…}
- 第二种形式：if-else语句，如：if(表达式) {…} else{…}
- 第三种形式：if - else if - else，如：if(表达式) {…} else if{…} else {…}

课堂练习：

要求：从键盘输入两个整数，输出其中的较大数，使用if-else语句完成。

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int number1,number2,max;
    printf("请输入两个数: \n");
    scanf("%d%d",&number1,&number2);
    if (number1 > number2) {
        max = number1;
    }else{
        max = number2;
    }
    printf("%d和%d中的较大数为:%d\n",number1,number2,max);
    return 0;
}
```

注：对于if-else语句，是**二选一结构**，即执行if后的语句块，则不执行else块，反之不执行if语句则必然执行else语句块。

- if - else if - else :

if - else if - else 为多选一的嵌套结构，测试多个表达式。一旦满足某个表达式，则执行其后代码后，if选择终止。

```
if (expression)
    statement1
else if (expression)
    statement2
else if (expression)
    statement3
else if (expression)
    statement4
else
    statement5//处理一些意外情况，错误检验。
```

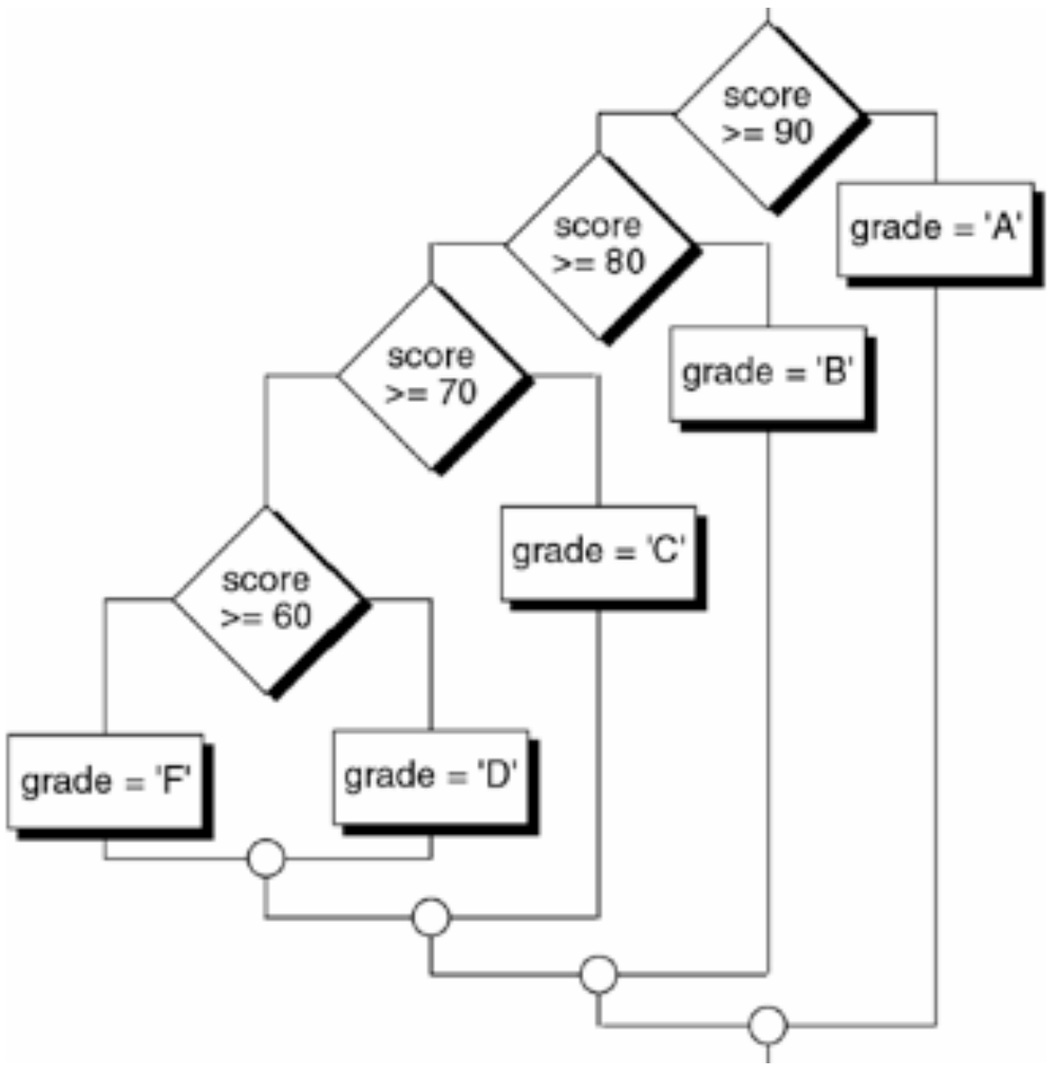
示例代码：

要求：用户要从键盘上读入一个100以内正整型的成绩给score,判断属于哪个成绩等级,并输出最终的成绩结果。

```c
#include <stdio.h>
int main(int argc, const char * argv[]) {
    int grade;
    char letter_grade;
    printf("Please enter a mark out of 100: ");//标准输出
    scanf("%d",&grade);//标准输入,等待用户输入,把键盘输入的值保存到grade变量。
    if ( grade >= 80 )
        letter_grade = 'A';
    if ( grade >= 70 && grade < 80 )
        letter_grade = 'B';
    if ( grade >= 60 && grade < 70 )
        letter_grade = 'C';
    if ( grade >= 50 && grade < 60 )
        letter_grade = 'D';
    if (grade < 50)
        letter_grade = 'F';
    printf("A mark of %d is an %c\n", grade, letter_grade);
    return 0;
}
```

代码分析:

上述程序全部使用if语句进行判断处理，效率低，当用户输入某一数值后，程序会对每一个if语句进行判断，并不会因为满足某个if语句后，执行完该if语句块的代码后停止if判断，如果对于大量的数据处理，效率会大大降低。



优化处理：使用多个else if来替换if语句。

```
#include <stdio.h>
int main(int argc, const char * argv[]){
    int score = 0;
    char letter_grade = '\0';
    printf("Please an integer mark out of 100: ");
    scanf("%d", &score);
    if (score > 100 || score <0)
        printf(" I'm afraid you have made an error in input \n");
    else if ( score >= 90 )//一旦某个else if条件满足了，在执行完这个else if
之后的代码后，后面的else if条件将不再进行判断，终止了整个嵌套的else if结构。
        letter_grade = 'A';
    else if ( score >= 80 )
        letter_grade = 'B';
    else if ( score >= 70 )
        letter_grade = 'C';
    else if ( score >= 60 )
        letter_grade = 'D';
    else
        letter_grade = 'F';
    if (score >= 0 && score <= 100)
        printf(" A grade of %d is an %c\n ", score, letter_grade);
    return 0;
}
```

3.2 switch选择语句

C语言还提供了另外一种用于多分支选择的switch语句。其一般形式为：

```
switch(表达式)
{
    case 常量表达式1：语句1;
    case 常量表达式1：语句1;
    …
    case 常量表达式n：语句n;
    default：默认语句;
}
```

其语义是：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switch块外的语句。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。

```c
#include <stdio.h>
int main( ){
    char letter_grade;
    printf("Enter your letter grade and press return: ");
    scanf("%c", &letter_grade); //标准输入
    switch (letter_grade){
        case 'A':
            printf("Your mark is between 80 and 100\n");
            break;
        case 'B':
            printf(" Your mark is between 70 and 79\n");
            break;
        case 'C':
            printf("Grade is between 60 and 69\n");
            break;
        case 'D':
            printf("Grade is between 50 and 59\n");
            break;
        case 'F':
            printf("Grade is below 50\n");
            break;
        default: //默认标签
            printf("You have typed in an illegal letter\n");
    }
    return 0;
}
```

代码分析:

上述程序全部使用switch结构，重新实现嵌套的if-else结构。用表达式letter_grade的值，和case后面的标签进行比较。当letter_grade的值与case后的常量表达式相等时，则执行该case后的语句，并且遇到break关键字后，switch语句退出。

在使用switch语句时还应注意以下几点:

(1) switch中的表达式的值只能是int、char、枚举类型，对于其它类型需使用if语句。

(2) 在case后的各常量表达式的值不能相同，否则会出现错误。

(3) break关键字是用于跳出switch选择结构。

(4) default语句是当所有的case都不能与switch中的表达式的值相匹配时执行。

四、循环结构

循环结构是程序中一种很重要的控制结构。其特点是在给定条件成立时,反复执行某个程序段,直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C语言提供了多种循环语句,可以组成各种不同形式的循环结构。

C语言的循环语句主要有三种形式:

- 第一种形式：for循环语句
- 第二种形式：while循环语句
- 第三种形式：do-while循环语句

4.1 for循环语句

for语句比while语句和do-while都要灵活，是一种功能更大、更常用的循环语句，它的一般语法格式为：

```
for（表达式1;表达式2;表达式3)
{
    循环体代码
}
```

其中，表示式可以省略，但是分号不可省略。

表达式1：一般为赋值表达式，给控制变量赋初值；int i = 0;

表达式2：关系表达式或逻辑表达式，循环控制条件；i < 5;

表达式3：一般为赋值表达式，给控制变量增量或减量。i++;

for循环流程图：

执行过程如右图所示。

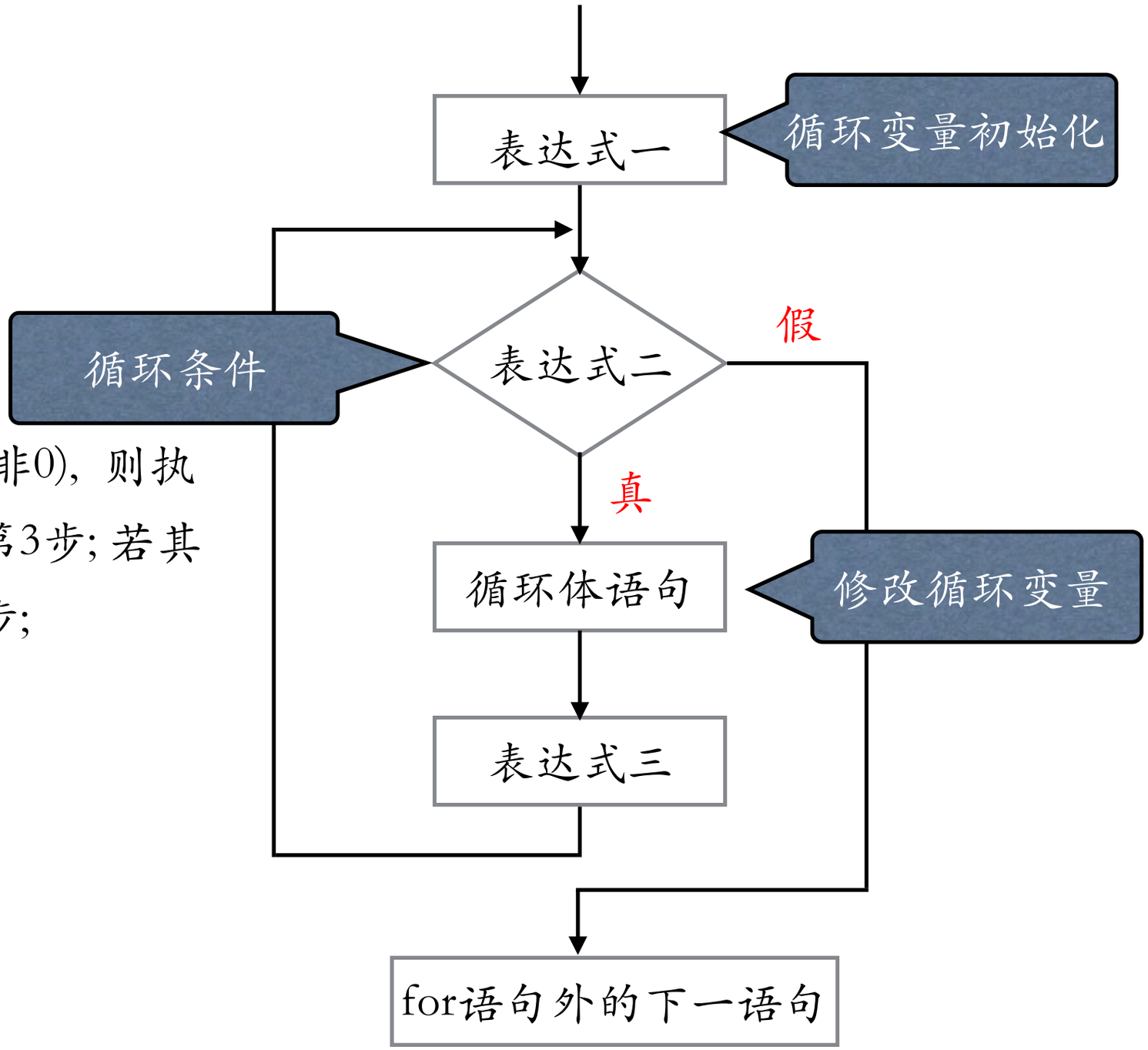
(1) 先求解表达式1;

(2) 再求解表达式2,若其值为真(非0), 则执行循环体语句,然后执行下面的第3步; 若其值为假(0), 则结束循环 转到第5步;

(3) 求解表达式3;

(4) 转回第2步,继续判断条件;

(5) 循环结束,执行for语句外的下一语句。



示例代码：

要求：求sum=1+2+3+4+...+100的和，使用for完成；

```
#include <stdio.h>
int main(int argc, const char * argv[])
{
    int i,sum=0; /*定义和初始化变量*/
    for(i=1;i<=100;i++)/*for(循环控制变量初始化;循环条件判段;循环变量改变)*/
    {
        sum=sum+i; /*求和语句*/
    }
    printf("sum = %d\n",sum);/*输出累加和*/
    return 0;
}
```

程序运行结果如下：

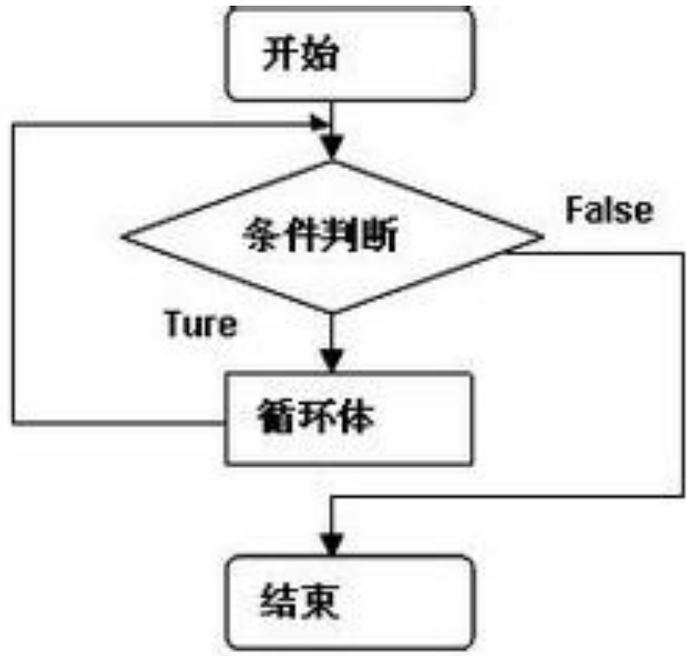
```
sum = 5050
```

4.2 while循环

循环语句，程序设计中一种基本循环模式。当满足条件时进入循环，不满足跳出。while语句的一般表达式为：

```
while（表达式）
{
    循环体
}
```

每次执行循环体前都要先对条件表达式进行判断。



示例代码:

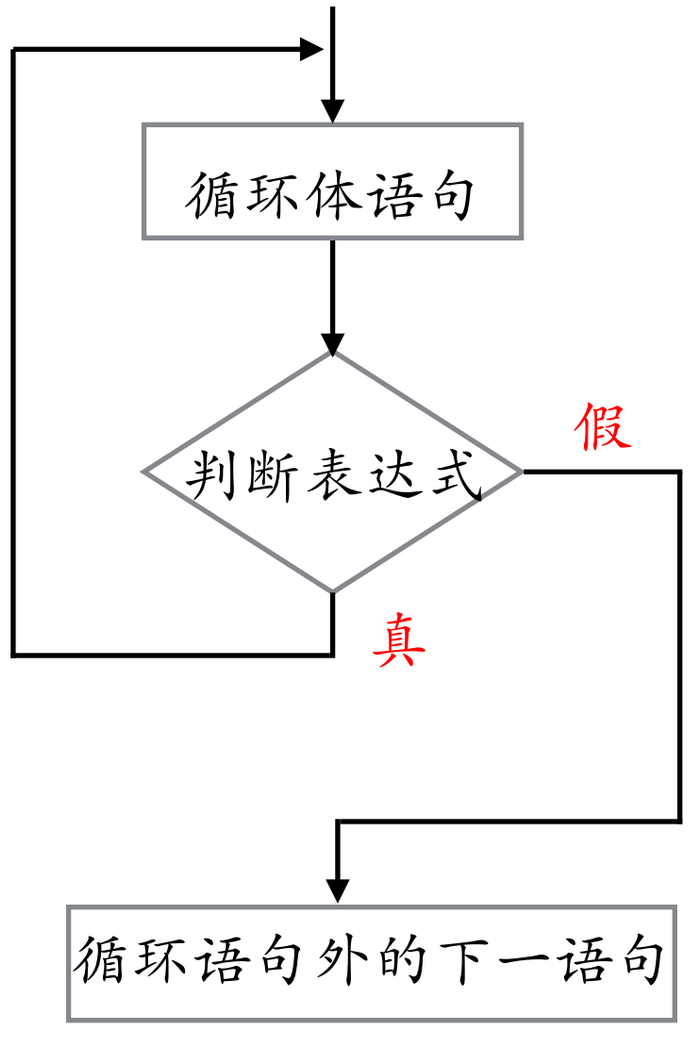
```c
#include <stdio.h>
int main(int argc, const char * argv[]){
    int count, sum, anInteger;
    printf("Enter the integers and terminate with negtive number\n");
    count = 0;
    sum = 0;
    printf("Enter number %d:", count+1);
    scanf("%d", &anInteger);
    while (anInteger >= 0)//判断条件表达式
    {
        sum += anInteger; //对所有的输入数进行求和
        count++;  //计数器
        printf("Enter number %d:", count+1);
        scanf("%d", &anInteger); //修改控制变量的值
    }
    if (count != 0)
        printf("The average is %f\n", sum / (double) count); //防止除0
    else
        printf("You entered no numbers\n");
    return 0;
}
```

4.3 do-while循环

do-while 循环是 while 循环的变体。在检查条件是否为真之前，该循环首先会执行一次代码块，然后检查条件是否为真，如果条件为真的话，就会重复这个循环。它的一般格式为:

```
do{
    循环体
}while（表达式）;
```

do-while循环是先执行一次循环体代码，再判断条件，即循环体代码至少被执行一次。



五、break和continue

5.1 break语句

break语句通常只用于开关语句switch和循环语句中。其般形式为:

```
break;
```

break语句的特点:

(1) 用于开关语句switch中:

可使程序跳出switch结构,继续执 switch之后的语句。

(2) 用于循环语句中:

可使程序终止循环,继续执行循环之后的语句。如果有多层循环，只跳出break所在这一层。通常break语句与if语句一起使用: 即满足某个条件时跳出循环。

示例代码：

求n!。（从键盘输入一个正整数n,求这个数的阶乘。）

```
#include <stdio.h>
int main(int argc, const char * argv[])
{
     int n,i;
     long int n_factorial = 1;
     printf("Enter integer: ");
     scanf("%d", &n);
     for (i=1; ;i++)
     {
          n_factorial *= i;
          if(i==n) /* if语句判断,当i的值与n的值相等时*/

              break; /*执 break语句,跳出内层循环*/

     }

     printf(" %d of factorial = %lu\n", n, n_factorial);
    return 0;
}
```

5.2 continue语句

continue: 用于循环体内部，作用是结束本次循环，开始执行新的循环。

示例代码:

```
#include <stdio.h>
int main(int argc, const char * argv[]){
    int i = 0;
    int array[3];
    int num;
    while (i < 3){
        printf( "\nInput the next number: ");
        scanf( "%d", &num );
        if ( num < 10 || num > 20 )
            continue;
        array[i] = num;
        i++;
    }
    for ( i = 0; i < 3; i++ ){
        printf( "The %d value: %d\n", i, array[i] );
    }
    return 0;
}
```

The End

渥瑞达·教学部