第 2 章　程序设计语言基础

大纲要求：

- 汇编、编译、解释系统的基础知识和基本工作原理。
- 程序设计语言的基本成分——数据、运算、控制和传输以及过程(函数)调用。
- 各类程序设计语言的主要特点和适用情况。

2.1　程序设计基础知识

2.1.1　考点辅导

2.1.1.1　程序设计语言的基本概念

1．低级语言和高级语言

1)　低级语言

通常称机器语言和汇编语言为低级语言。机器语言是指用 0、1 字符串组成的机器指令序列，是最基本的计算机语言；汇编语言是指用符号表示指令的语言。

2)　高级语言

高级语言是从人类的逻辑思维角度出发、面向各类应用的程序语言，抽象程度大大提高，需要经过编译成特定机器上的目标代码才能执行。这类语言与人们使用的自然语言比较接近，大大提高了程序设计的效率。

2．编译程序和解释程序

尽管人们可以借助高级语言和计算机进行交互，但是计算机仍然只能理解和执行由 0、1 序列构成的机器语言，因此高级程序语言需要翻译，担任这一任务的程序称为语言处理程序。用某种高级语言或汇编语言编写的程序称为源程序，源程序不能直接在计算机上执行。如果源程序是使用汇编语言编写的，则需要一个称为汇编程序的翻译程序将其翻译成目标程序后才能执行。如果源程序是使用某种高级语言编写的，则需要相应的解释程序或编译程序对其进行翻译，然后才能在机器上执行。

解释程序也称为解释器，它或者直接解释执行源程序，或者将源程序翻译成某种中间表示形式后再执行；而编译程序(编译器)则是将源程序翻译成目标语言程序，然后在计算机上执行目标程序。

3．程序设计语言的定义

下面介绍关于程序设计语言的定义。

(1)　语法。由程序设计语言的基本符号组成程序中的各个语法成分(包括程序)的一组规

则，其中由基本符号构成的符号(单词)书写规则称为词法规则，由符号(单词)构成语法成分的规则称为语法规则。程序语言的语法可通过形式语言进行描述。

(2) 语义。程序语言中按语法规则构成的各个语法成分的含义，可分为静态语义和动态语义。

(3) 语用。表示构成语言的各个记号和使用者的关系，涉及符号的来源、使用和影响。

(4) 语境。理解和实现程序设计语言的环境，包括编译环境和运行环境。

4．程序设计语言的分类

1) 命令式程序设计语言

命令式程序设计语言是基于动作的语言，在这种语言中，计算被看作动作的序列。命令式语言族开始于 FORTRAN、PASCAL 和 C 语言，体现了命令式程序设计的关键思想。

2) 面向对象的程序设计语言

(1) 对象。对象是指人们要进行研究的任何事物，它具有状态和操作。面向对象的语言把状态和操作封装于对象实体之中，并提供一种访问机制。用户只能通过向允许公开的操作提出要求，才能查询和修改对象的状态。

(2) 类。类是面向对象语言必须提供的、由用户定义的数据类型，它将具有相同状态、操作和访问机制的多个对象抽象成一个对象类。在定义类以后，属于这种类的一个对象被称为类实例或类对象。

(3) 继承。继承是面向对象语言的另一个基本要素。继承实现了一般与特殊的关系，解决了软件的可重用性和可扩充性的问题。

3) 函数式程序设计语言

函数式程序设计语言是一类以λ-演算为基础的语言。该语言的代表是 LISP 语言，其中大量使用了递归。

函数是一种对应规则(映射)，它是定义域中的每个元素和值域中唯一的元素相对应。函数可以看成一种程序，其输入就是定义在左边括号中的变量，可以将输入组合起来产生一个规则，组合过程中也可以使用其他函数或函数本身。这种用函数和表达式建立程序的方法就是函数式程序设计。函数型程序设计语言的优点之一就是表达式中出现的任何函数都可以用其他函数来代替，只要这些函数调用产生相同的值。

4) 逻辑型程序设计语言

逻辑型程序设计语言是一类以形式逻辑为基础的语言。该语言的代表是建立在关系理论和一阶谓词理论基础上的 Prolog 语言。Prolog 语言具有很强的推理功能，适用于书写自动定理证明、专家系统以及自然语言理解等问题的程序。

2.1.1.2 程序设计语言的基本成分

1．数据成分

程序语言的数据成分是指一种程序语言的数据类型。

1) 常量和变量

按照程序运行时数据的值能否改变，将数据分为常量和变量。程序中的数据对象可以具有左值和(或)右值，左值是指存储单元(或地址、容器)，右值是指具体值(或内容)。变量具有左值和右值，在程序运行过程中其右值可以改变；常量只有右值，在程序运行过程中

其右值不能改变。

2)　全局量和局部量

按数据的作用域范围，数据可分为全局量和局部量。系统为全局变量分配的存储空间在程序运行的过程中一般是不改变的，而为局部变量分配的存储单元是动态改变的。

3)　数据类型

按照数据组织形式的不同可将数据分为基本类型、用户定义类型、构造类型及其他类型。C(C++)的数据类型如下。

- 基本类型：整型(int)、字符型(char)、实型(float、double)和布尔类型(bool)。
- 特殊类型：空类型(void)。
- 用户定义类型：枚举类型(enum)。
- 构造类型：数组、结构体和共用体。
- 指针类型：type *。
- 抽象数据类型：类类型。

其中，布尔类型和类类型是 C++在 C 语言的基础上扩充的。

2．运算成分

程序语言的运算成分是指允许使用的运算符号及运算规则。大多数高级程序语言的基本运算可以分成算术运算、关系运算和逻辑运算，有些语言还提供位运算。运算符号的使用与数据类型密切相关。为了确保运算结果的唯一性，运算符号要规定优先级和结合性，必要时还要使用圆括号。

3．控制成分

控制成分指明语言允许表述的控制结构，程序员使用控制成分来构造程序中的控制逻辑。

1)　顺序结构

在顺序结构中，计算过程从所描述的第一个操作开始，按顺序依次执行后续的操作，直到执行完序列的最后一个操作。顺序结构内也可以包含其他控制结构。

2)　选择结构

选择结构提供了在两种或多种分支中选择执行其中一个分支的逻辑。基本的选择结构是指定一个条件 P，然后根据条件的成立与否决定控制流走计算 A 还是走计算 B，从两个分支中选择一个执行。选择结构中的计算 A 或计算 B 还可以包含顺序、选择和重复结构。程序语言中通常还提供简化了的选择结构，也就是没有计算 B 的分支结构。

3)　循环结构

循环结构描述了重复计算的过程，通常包括 3 个部分，即初始化、需要重复计算的部分和重复的条件。其中初始化部分有时在控制的逻辑结构中不进行显式表示。循环结构主要有两种形式，即 while 型重复结构和 do-while 型重复结构。

4)　C(C++)语言提供的控制语句

(1)　复合语句。复合语句用于描述顺序控制结构。复合语句是一系列用“{”和“}”括起来的声明和语句，其主要作用是将多条语句组成一个可执行单元。复合语句是一个整体，要么全部执行，要么一条语句也不执行。

(2) if 语句和 switch 语句。这两种语句用于实现选择结构。

① if 语句实现的是双分支的选择结构，其一般形式如下：

```
if(表达式)语句 l;else 语句 2;
```

其中，语句 1 和语句 2 可以是任何合法的 C(C++)语句，当语句 2 为空语句时，可以简化为

```
if(表达式) 语句;
```

使用 if 语句时，需要注意的是 if 和 else 的匹配关系。C 语言规定，else 总是与离它最近的尚没有 else 与其匹配的 if 相匹配。

② switch 语句描述了多分支的选择结构，其一般形式如下：

```
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;
}
```

执行 switch 语句时，首先计算表达式的值，然后用所得的值与列举的常量表达式值依次比较，若任一常量表达式都不能与所得的值相匹配，则执行 default 的“语句序列 n+l”，然后结束 switch 语句。

表达式可以是任何类型，常用的是字符型或整型表达式。多个常量表达式可以共用一个语句组。语句组可以包括任何可执行语句，且无须用“{”和“}”括起来。

(3) 循环语句。C(C++)语言提供了 3 种形式的循环语句，用于描述循环计算的控制结构。

① while 语句。while 语句描述了先判断条件再执行循环体的控制结构，其一般形式如下：

```
while(条件表达式) 循环体语句;
```

② do-while 语句。do-while 语句描述了先执行循环体再判断条件的控制结构，其一般格式如下：

```
do
    循环体语句;
while(条件表达式);
```

③ for 语句。for 语句的基本格式如下：

```
for(表达式 1;表达式 2;表达式 3) 循环体语句;
```

可用 while 语句等价地表示为

```
表达式 1;
while(表达式 2){
    循环体语句;
    表达式 3;
}
```

4．函数

函数是程序模块的主要成分，它是一段具有独立功能的程序。函数的使用涉及 3 个概念，即函数定义、函数声明和函数调用。

(1) 函数定义。包括函数首部和函数体两个部分。函数的定义描述了函数做什么和怎么做。

(2) 函数声明。函数应该先声明后引用。函数声明定义了函数原型。声明函数原型的目的在于告诉编译器传递给函数的参数个数、类型以及函数返回值的类型，参数表中仅需要依次列出函数定义中的参数类型。函数原型可以使编译器检查源程序中对函数的调用是否正确。

(3) 函数调用。当需要在一个函数(称为主调函数)中使用另一个函数(称为被调函数)实现的功能时，便以函数名字进行调用，称为函数调用。调用函数和被调用函数之间交换信息的方法主要有两种：一种是由被调用函数把返回值返回给主调函数；另一种是通过参数带回信息。函数调用时实参和形参间交换信息的方法有传值调用和引用调用两种。

① 传值调用(Call by Value)。若实现函数调用时实参向形式参数传递相应类型的值(副本)，则称为传值调用。这种方式下形式参数不能向实际参数传递信息。在 C 语言中，要实现被调用函数对实际参数的修改，必须用指针作形参。即调用时需要先对实参进行取地址运算，然后将实参的地址传递给指针形参，本质上仍属于传值调用。这种方式实现了间接内存访问。

② 引用调用(Call by Reference)。引用是 C++中增加的数据类型，当形式参数为引用类型时，形参名实际上是实参的别名，函数中对形参的访问和修改实际上就是针对相应实际参数所作的访问和改变。

2.1.2　典型例题分析

例 1　以下关于解释程序和编译程序的叙述中，正确的是 (20) 。(2013 年上半年试题 20)

(20) A. 编译程序和解释程序都生成源程序的目标程序

B. 编译程序和解释程序都不生成源程序的目标程序

C. 编译程序生成源程序的目标程序，解释程序则不然

D. 编译程序不生成源程序的目标程序，而解释程序反之

解析：编译程序也称编译器，将源程序翻译成目标语言程序，然后在计算机上运行目标程序。虽然执行效率高，但编写出来的程序可读性很差，且难以修改和维护。

解释程序：也称解释器，它或者直接解释执行源程序，或者将源程序翻译成某种中间代码后再加以执行。

答案：C

例 2　以下关于传值调用与引用调用的叙述中，正确的是 (21) 。(2013 年上半年试题 21)

① 在传值调用方式下，可以实现形参和实参间双向传递数据的效果

② 在传值调用方式下，实参可以是变量，也可以是常量和表达式

③ 在引用调用方式下，可以实现形参和实参间双向传递数据的效果

④ 在引用调用方式下，实参可以是变量，也可以是常量和表达式

(21) A. ①③　　B. ①④　　C. ②③　　D. ②④

解析：传值调用最显著的特征是被调用的函数内部对形参的修改不影响实参的值。传值调用的参数传递和被调用函数内对参数的使用主要按下述原则：函数定义时形参被当作局部变量看待，在函数被调用时为形参分配存储单元；调用函数前，首先计算实参的值，调用时将实参的值放入形参的存储单元；被调用函数内部对形参单元中的数据进行直接访问。

引用调用是将实参的地址传递给形参，使得形参的地址就是对应实参的地址。引用调用的参数传递和被调用函数内对参数的使用主要按下述原则处理：函数定义时形参被当作局部变量看待，在函数被调用时为形参分配存储单元；调用时将实参的地址放入形参的存储单元；被调用函数内部对形参单元中的数据(地址)进行间接访问。

答案：C

例 3　可用于编写独立程序和快速脚本的语言是 (20) 。(2012 年下半年试题 20)

(20) A．Python　　B．Prolog　　C．Java　　D．C#

解析：Python 是一种面向对象的解释型程序设计语言，可用于编写独立程序、快速脚本和复杂应用的原型。Python 也是一种脚本语言，它支持对操作系统底层的访问。

Prolog 是一种逻辑型语言。Prolog 程序是一系列事实、数据对象或事实间的具体关系和规则的集合。Prolog 有很强的推理功能，适用于书写自动定理证明、专家系统、自然语言理解等问题的程序。

Java 是一种面向对象的程序设计语言，能开发应用在 Internet 上且具有软、硬件独立性和交互能力的程序。Java 语言的程序可以一次编写而到处运行。

C#是微软公司发布的一种面向对象的、运行于.NET Framework 之上的高级程序设计语言。C#看起来与 Java 有着惊人的相似：它包括了如单一继承、接口、与 Java 几乎同样的语法和编译成中间代码再运行的过程。但是 C#与 Java 有着明显的不同，它借鉴了 Delphi 的一个特点，与 COM(组件对象模型)是直接集成的，而且它是微软公司.NET Windows 网络框架的主角。

答案：A

例 4　将高级语言源程序翻译成目标程序的是 (48) 。(2012 年下半年试题 48)

(48) A．解释程序　　B．编译程序　　C．链接程序　　D．汇编程序

解析：编译程序的功能是把某高级语言书写的源程序翻译成与之等价的目标程序。解释程序是另一种语言处理程序，在词法、语法和语义分析方面与编译程序的工作原理基本相同，但在运行用户程序时，它直接执行源程序或源程序的中间表示形式。解释程序不产生源程序的目标程序，这是它和编译程序的主要区别。

答案：B

例 5　以下关于程序设计语言的叙述中，错误的是 (20) 。(2015 年上半年试题 20)

(20) A．程序设计语言的基本成分包括数据、运算、控制和传输等

B．高级程序设计语言不依赖于具体的机器硬件

C．程序中局部变量的值在运行时不能改变

D．程序中常量的值在运行时不能改变

解析：变量具有左值和右值，在程序运行过程中，局部变量的右值可以改变。

答案：C

例 6　函数(过程)调用时，常采用传值与传地址两种方式在实参和形参间传递信息。以下叙述中，正确的是__(50)__。(2012 年上半年试题 50)

(50) A. 在传值方式下，将形参的值传给实参，因此，形参必须是常量或变量

B. 在传值方式下，将实参的值传给形参，因此，实参必须是常量或变量

C. 在传地址方式下，将形参的值传给实参，因此，形参必须有地址

D. 在传地址方式下，将实参的值传给形参，因此，实参必须有地址

解析：形式参数就是过程定义中函数名后括号中所带的参数；实际参数是在调用点表示向被调用过程传递的数据。在函数调用时，数据传递的方向是从实参到形参。只是采用传值传递方式时，传递的是数值，这个数值只要是确定的即可，可以是常理、变量或表达式等。而采用传址传递方式时，传递的是地址，因此实参必须有地址。

答案：D

例 7　编译器和解释器是两种基本的高级语言处理程序。编译器对高级语言源程序的处理过程可以划分为词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成等阶段，其中，__(20)__并不是每个编译器都必需的，与编译器相比，解释器__(21)__。(2015 年下半年试题 20、21)

(20) A．词法分析和语法分析　　B．语义分析和中间代码生成

C．中间代码生成和代码优化　　D．代码优化和目标代码生成

(21) A．不参与运行控制，程序执行的速度慢

B．参与运行控制，程序执行的速度慢

C．参与运行控制，程序执行的速度快

D．不参与运行控制，程序执行的速度快

解析：在编译过程中中间代码的生成与优化不是必需的，但用中间代码有很多的好处，最重要的是两点：①便于实现优化，使最终代码的质量更高；②通过中间代码实现前后级分离，在多系统、多语言开发时，可大幅提高整体开发效率，减少开发成本、缩短开发周期。所以实际的编译系统多数都会使用中间代码。

在解释器上运行程序比直接运行编译过的代码要慢，是因为解释器每次都必须去分析并转译它所运行到的程序行，而编译过的程序直接运行即可。

答案：(20) C　(21) B

例 8　若 C 程序的表达式中引用了未赋初值的变量，则__(20)__。(2011 年下半年试题 20)

(20) A. 编译时一定会报告错误信息，该程序不能允许

B. 可以通过编译并运行，但运行时一定会报告异常

C. 可以通过编译，但链接时一定会报告错误而不能运行

D. 可以通过编译并运行，但运行结果不一定是期望的结果

解析：全局变量和静态局部变量在定义时如果没有初始化，编译时会自动初始化为 0；而普通的局部变量，如果没有初始化，则其值是一个随机数。在 C 程序表达式中，只要引用的变量定义了，就可以通过编译并运行，但运行结果不一定是期望的结果。

答案：D

例 9 函数 t()、f()的定义如下所示，若调用函数 t 时传递给 x 的值为 5，并且调用函数 f()时，第一个参数采用传值(call by value)方式，第二个参数采用引用(call by reference)方式，则函数 t 的返回值为 (50) 。(2015 年下半年试题 50)

```
t(int x)
int a;
a=3*x+1;
f(x,a);
return a-x;
```

```
f(int r, int &s)
int x;
x=2*s+1; s=x+r;
r=x−1;
return;
```

(50) A．33　　B．22　　C．11　　D．负数

解析：在函数 t 中，执行语句 a=3*x+1，得 a=16；调用 f(x,a)时，将 x 的值 5、a 的值 16 传递给函数 f 的形参 r 和 s。由于 r 采用的是传值方式，函数调用后不会改变 x 的值；而参数 s 采用的是引用方式，函数调用后 a 的值发生改变。函数 f 执行完成后 a 的值变为 38，x 的值不变，为 5，因此函数 t 的返回值为 a−x= 38−5=33。

答案：A

例10 在引用调用方式下进行函数调用是将 (21) 。(2014年上半年试题21)

(21) A．实参的值传递给形参　　B．实参的地址传递给形参

C．形参的值传递给实参　　D．形参的地址传递给实参

解析：引用调用是把实参(如int a)的地址(&a)赋给形参(指针变量，比如*b,这时b=&a,即b指向变量a),如果*b(也即a对应的内存空间)发生变化，也就是变量a的值发生了变化。

答案：B

例 11 以下关于变量和常量的叙述中，错误的是 (20) 。(2010 年下半年试题 20)

(20) A．变量的取值在程序运行过程中可以改变，常量则不行

B．变量具有类型属性，常量则没有

C．变量具有对应的存储单元，常量则没有

D．可以对变量赋值，不能对常量赋值

解析：常量是在程序运行过程中值不可以改变的数据。根据数据的组织类型的不同，可以将数据分为基本数据类型、用户自定义数据类型、构造类型等。变量具有类型属性，常量也有数据类型，如整数常量、字符串常量等。

答案：B

例 12 下面 C 程序段中 count++语句执行的次数为 (64) 。(2010 年下半年试题 64)

```
for(int i=1;i<=11;i*=2)
    for(int j=1;j<=i;j++)
        count++;
```

(64) A．15　　B．16　　C．31　　D．32

解析：第 1 轮循环，i=1，count++执行 1 次，然后 i=2；第 2 轮循环，i=2，count++执行 2 次，然后 i=4；第 3 轮循环，i=4，count++执行 4 次，然后 i=8；第 4 轮循环，i=8，count++执行 8 次，然后 i=16，i>11，不满足循环条件，循环结束。可以计算 count++语句执行的次数为 1+2+4+8=15。

答案：A

例 13 程序的 3 种基本控制结构是 (33) 。(2010 年上半年试题 33)

(33) A．过程、子程序和程序　　B．顺序、选择和重复
　　C．递归、堆栈和队列　　D．调用、返回和跳转

解析：程序的 3 种基本控制结构是顺序结构、选择结构和重复结构。

答案：B

例 14　函数调用时，基本的参数传递方式有传值与传址两种，_(20)_。(2009 年上半年试题 20)

(20) A．在传值方式下，形参将值传给实参
　　B．在传值方式下，实参不能是数组元素
　　C．在传址方式下，形参和实参间可以实现数据的双向传递
　　D．在传址方式下，实参可以是任意的变量和表达式

解析：首先看 A 选项。在传值方式下，对应的实参和形参是两个独立的实体，占用不同的内存单元，调用函数时，系统把实参值复制一份给形参，便断开两者的联系，形参值的改变对实参无影响。因此，“传值”是单向的，只能由实参传递给形参。

B 选项，形参为传值方式下的简单变量，实参可以是与其同类型的常量、变量、数组元素或表达式。

C 选项，在传址方式下，函数调用时，系统将实参的地址传递给形参，即这时参数传递的不是数据本身，而是数据在内存中的地址。所以在被调用函数中，任何对形式参数的访问，都被认为是对形式参数的间接访问。实参与形参占用不同的存储单元，传递方式是双向的，形参值的改变将影响实参值。故 C 选项正确。

D 选项，形参为传址方式时，实参如果为常量或表达式，则传址无效，相当于传值方式。

答案：C

例 15　下面关于程序语言的叙述，错误的是_(22)_。(2009 年上半年试题 22)

(22) A．脚本语言属于动态语言，其程序结构可以在运行中改变
　　B．脚本语言一般通过脚本引擎解释执行，不产生独立保存的目标程序
　　C．PHP、JavaScript 属于静态语言，其所有成分可在编译时确定
　　D．C#、Java 语言属于静态语言，其所有成分可在编译时确定

解析：脚本是一种特定的描述性语言，是依据一定的格式编写的可执行文件，又称作宏或批处理文件。脚本通常可以由应用程序临时调用并执行。脚本语言一般都是以文本形式存在，类似于一种命令。有些程序，如 C、C++、Java 等则必须先经过编译，将源代码转换为二进制代码之后才可执行。而像 Perl、JavaScript、VBScript 等则不需要事先编译，只要利用合适的解释器便可以执行代码。

动态类型语言是指在运行期间才去做数据类型检查的语言。也就是说，在用动态类型的语言编程时，永远也不用给任何变量指定数据类型，该语言会在你第一次赋值给变量时，在内部将数据类型记录下来，不用编译即可运行。Python 和 Ruby 就是一种典型的动态类型语言，其他的各种脚本语言如 JavaScript 属于动态类型语言。静态类型语言的数据类型是在编译期间检查的，也就是说，在写程序时要声明所有变量的数据类型。C/C++是静态类型语言的典型代表，其他的静态类型语言还有 C#、Java 等。故错误的为选项 C。

答案：C

例16　在某C/C++程序中，整型变量a的值为0且应用在表达式“c=b/a”中，则最可能

发生的情形是 (50) 。(2014年上半年试题50)

(50) A．编译时报告有语法错误　　B．编译时报告有逻辑错误

C．运行时报告有语法错误　　D．运行时产生异常

解析：编译时a的值无法确定，表达式“ c=b/a”符合C/C++语言的语法逻辑，编译时不会报错。运行时，代入a的值，发生错误。

答案：D

2.1.3 同步练习

1. 程序设计语言一般都提供多种循环语句，如实现先判断循环条件再执行循环体的 while 语句和先执行循环体再判断循环条件的 do-while 语句。关于这两种循环语句，在不改变循环体的条件下，_____是正确的。

A．while 语句的功能可由 do-while 语句实现

B．do-while 语句的功能可由 while 语句实现

C．若已知循环体的次数，则只能使用 while 语句

D．循环条件相同时，do-while 语句的执行效率更高

2. 下列叙述中错误的是_____。

A．面向对象程序设计语言可支持过程化的程序设计

B．给定算法的时间复杂性与实现该算法所采用的程序设计语言无关

C．与汇编语言相比，采用脚本语言编程可获得更高的运行效率

D．面向对象程序设计语言不支持对一个对象的成员变量进行直接访问

2.1.4 同步练习参考答案

1．C

2．C

2.2 语言处理程序基础

2.2.1 考点辅导

2.2.1.1 汇编语言的基本原理

1．汇编语言

汇编语言是为特定的计算机或计算机系统设计的面向机器的符号化的程序设计语言。用汇编语言编写的程序称为汇编语言源程序。

汇编语言源程序由若干条语句组成。一个程序中可以有 3 类语句，即指令语句、伪指令语句和宏指令语句。

(1) 指令语句。又称为机器指令语句，汇编后能产生相应的机器代码，被 CPU 直接识别并执行相应的操作。指令语句可分为传送指令、算术运算指令、逻辑运算指令、移位指令、转移指令和处理机控制指令等。

(2) 伪指令语句。指示汇编程序在对源程序进行汇编时完成某些工作。与指令语句的区别是：伪指令语句经汇编后不产生机器代码，另外，伪指令语句所指示的操作是在源程序被汇编时完成的，而指令语句的操作必须在程序运行时完成。伪指令语句包括常数定义伪指令语句、存储定义伪指令语句、开始伪指令语句和结束伪指令语句。

(3) 宏指令语句。将多次重复使用的程序段定义为宏。宏的定义必须按照相应的规定进行，每个宏都有相应的宏名。

2．汇编程序

汇编程序的功能是将用汇编语言编写的源程序翻译成机器指令程序。它一般至少需要两次扫描源程序才能完成翻译过程。第一次扫描的主要工作是定义符号的值并创建一个符号表(ST)；第二次扫描的任务是产生目标程序。除了使用前一次扫描所产生的符号表(ST)外，还要使用机器指令表(MOT2)。在第二次扫描过程中，可执行汇编语句应被翻译成对应的二进制代码机器指令。这一过程涉及两个方面的工作：一是把机器指令助记符转换成二进制机器指令操作码，这可通过查找 MOT2 来实现；二是求出操作数区各操作数的值(用二进制表示)。

2.2.1.2　编译程序的基本原理

1．编译过程概述

编译程序的功能是把用高级语言书写的源程序翻译成与之等价的目标程序。编译过程划分成词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成 6 个阶段，实际的编译器可能会将其中的某些阶段结合在一起进行处理，比如说表格管理和出错处理与上述 6 个阶段都有联系。

1)　词法分析阶段

词法分析阶段的任务是对源程序从前到后(从左到右)逐个字符进行扫描，从中识别出一个个“单词”符号。“单词”符号是程序设计语言的基本语法单位，如关键词、标识符等。词法分析程序输出的“单词”常常采用二元组的方式，即单词类别和单词自身的值。

2)　语法分析阶段

语法分析的任务是在词法分析的基础上，根据语言的语法规则将单词符号序列分解成各类语法单位，如“表达式”“语句”和“程序”等。

词法分析和语法分析本质上都是对源程序的结构进行分析。

3)　语义分析阶段

语义分析阶段主要是审查源程序是否存在语义错误，并收集类型信息供后面的代码生成阶段使用，只有语法和语义都正确的源程序才能翻译成正确的目标代码。语义分析的一个主要工作是进行类型分析和检查。

4)　中间代码生成阶段

中间代码是一种结构简单且含义明确的记号系统，可以有多种形式。中间代码的设计原则主要有两点：一是容易生成；二是容易被翻译成目标代码。中间代码生成阶段的工作

就是根据语义分析的输出生成中间代码。

语义分析和中间代码生成所依据的是语言的语义规则。

5) 代码优化阶段

代码优化阶段的任务是对前阶段产生的中间代码进行变换或进行改造，目的是使生成的目标代码更为高效，即省时间和省空间。优化过程可以在中间代码生成阶段进行，也可以在目标代码生成阶段进行。

6) 目标代码生成阶段

目标代码生成阶段的任务是把中间代码变换成特定机器上的绝对指令代码、可重定位的指令代码或汇编指令代码。这是编译的最后阶段，它的工作与具体的机器密切相关。

7) 符号表管理

符号表管理阶段的任务是在符号表中记录源程序中各个符号的必要信息，以辅助语义的正确性检查和代码生成。符号表的建立可以始于词法分析阶段，也可以放到语法分析阶段，但符号表的使用有时会延续到目标代码的运行阶段。

8) 出错处理

用户编写的源程序中的错误大致可分为静态错误和动态错误。动态错误也称为动态语义错误，指程序中包含的逻辑错误。静态错误是指编译阶段发现的程序错误，可分为语法错误和静态语义错误。出错处理程序的任务包括检查错误、报告出错信息、排错、恢复编译工作。

2. 文法和语言的形式描述

1) 文法的定义

描述语言语法结构的形式规则称为文法。文法 G 是一个四元组，可表示为 $G=(V_N, V_T, P, S)$，其中 V_T 是一个非空有限集，其中的每个元素称为一个终结符；V_N 是一个非空有限集，其每个元素称为非终结符。$V_N \cap V_T = \varnothing$。$P$ 是产生式的有限集合，每个产生式是形如 $\alpha \rightarrow \beta$ 的规则，其中 α 称为产生式的左部，β 称为产生式的右部。$S \in V_N$，称为开始符号，它至少要在一条产生式中作为左部出现。

2) 文法的分类

乔姆斯基把文法分成 4 种类型，即 0 型、1 型、2 型和 3 型。

0 型文法也称为短语文法，其能力相当于图灵机。

(1) 1 型文法也称为上下文有关文法，这种文法意味着对非终结符的替换必须考虑上下文，并且一般不允许替换成 ε 串，此文法对应于线性有界自动机。

(2) 2 型文法是上下文无关文法，对非终结符的替换无须考虑上下文，它对应于下推自动机。

(3) 3 型文法等价于正规式，因此也称为正规文法或线性文法，它对应于有限状态自动机。

3) 句子和语言

设有文法 $G=(V_N, V_T, P, S)$

(1) 推导和直接推导。从文法的开始符号 S 出发，反复使用产生式，将产生式左部的非终结符替换为右部的文法符号序列，直至产生一个终结符的序列时为止。若有产生式 $\alpha \rightarrow \beta \in P$，$\gamma, \delta \in V^*$，则 $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ 称为文法 G 中的一个直接推导。

(2)　直接归约和归约。若文法 G 中有一个直接推导$\alpha \Rightarrow \beta$，则称α是β的一个直接归约；若文法 G 中有一个推导$\gamma \overset{*}{\underset{G}{\Rightarrow}} \delta$，则称$\gamma$是$\delta$的一个归约。

(3)　句型和句子。若文法 G 的开始符号为 S，那么，从开始符号 S 能推导出的符号串称为文法的一个句型，即α 是文法 G 的一个句型，当且仅当有以下推导 $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$，$\alpha \in V^*$，若 X 是文法 G 的一个句型，且 $X \in V_T^*$，则称 X 是文法 G 的一个句子。

(4)　语言。从文法 G 的开始符号出发，所能推导出的句子的全体称为文法 G 产生的语言，记为 $L(G)$。

(5)　文法的等价。若文法 G_1 与文法 G_2 产生的语言相同，即 $L(G_1)=L(G_2)$，则称这两个文法是等价的。

3．词法分析

1)　正规表达式和正规集

对于字母表Σ，其上的正规表达式(也称正则表达式，简称正规式)及其表示的正规集可以递归定义如下。

(1)　ε 是一个正规式，它表示集合 $L(\varepsilon)=\{\varepsilon\}$。

(2)　若 a 是Σ上的字符，则 a 是一个正规式，它所表示的正规集为$\{a\}$。

(3)　若正规式 r 和 s 分别表示正规集 $L(r)$和 $L(s)$，则

① $r|s$ 是正规式，表示集合 $L(r) \cup L(s)$。

② $r \cdot s$ 是正规式，表示集合 $L(r)L(s)$。

③ r^*是正规式，表示集合$(L(r))^*$。

④ (r)是正规式，表示集合 $L(r)$。

仅由有限次地使用上述 3 个步骤定义的表达式才是 Σ 上的正规式，其中运算符“|”“·”“*”分别称为“或”“连接”和“闭包”。若两个正规式表示的正规集相同，则认为两者等价。

2)　有限自动机

有限自动机是一种识别装置的抽象概念，它能够正确地识别正规集。

(1)　确定的有限自动机。

一个确定的有限自动机(DFA)是个五元组：(S, Σ, f, s_0, Z)，其中：

① S 是一个有限集，其每个元素称为一个状态。

② Σ是一个有限字母表，其每个元素称为一个输入字符。

③ f 是从 $S \times \Sigma \to S$ 上的单值部分映像。

④ $s_0 \in S$ 是唯一的一个开始状态。

⑤ Z 是非空的终止状态集合。

一个 DFA 可以用两种直观的方式表示，即状态转换图和状态转换矩阵。状态转换图简称为转换图，它是一个有向图。DFA 中的每个状态对应转换图中的一个节点，DFA 中的每个转换函数对应图中的一条有向弧，若转换函数为 $f(A, a)=Q$，则该有向弧从节点 A 出发，进入节点 Q，字符 a 是弧上的标记。状态转换矩阵可以用一个二维数组 M 表示，矩阵元素的行下标表示状态，列下标表示输入字符，$M[A,a]$的值是当前状态为 A、输入为 a 时应转换

到的下一状态。在转换矩阵中，一般以第一行的行下标所对应的状态作为初态，而终态则需要特别指出。

(2) 不确定的有限自动机。

一个不确定的有限自动机(NFA)也是一个五元组，它与确定的有限自动机的区别如下。

① f 是从 $S\times\Sigma\to 2^S$ 上的映像。对于 S 中的一个给定状态及输入符号，返回一个状态的集合。

② 有向弧上的标记可以是ε。

显然，DFA 是 NFA 的特例。

实际上，对于每个 NFA M，都存在一个 DFA N，且 $L(M)=L(N)$。

对于任何两个有限自动机 M_1 和 M_2，如果 $L(M_1)=L(M_2)$，则称 M_1 和 M_2 是等价的。

3) NFA 到 DFA 的转换

设 NFA $N=(S,\Sigma,f,s_0,Z)$，与之等价的 DFA $M=(S',\Sigma,f',q_0,Z')$，用子集法将非确定的有限自动机确定化的算法步骤如下。

(1) 求出 DFA M 的初态 q_0，此时 S'仅含初态 q_0，并且没有标记。

(2) 对于 S'中尚未标记的状态 $q_i=\{s_{i1},s_{i2},\mathrm{L},s_{im}\}$和 $s_{ij}\in S(j=1,2,\mathrm{L},m)$进行下述处理。

① 标记 q_i。

② 对于每个 $a\in\Sigma$，令 $T=f(s_{i1},s_{i2},\mathrm{L},s_{im},a)$，$q_j=\varepsilon\_\mathrm{CLOSURE}(T)$。

③ 若 q_i 尚不在 S'中，则将 q_j 作为一个未加标记的新状态添加到 S'，并把状态转换函数 $f'(q_i,a)=q_j$ 添加到 DFA M。

(3) 重复步骤(2)，直到 S'中不再有未标记的状态时为止。

(4) 令 $Z'=\{q\,|\,q\in S'\text{且}q\cap Z\neq\varnothing\}$。

注：若 I 是 NFA N 的状态集合的一个子集，其中 $\varepsilon\_\mathrm{CLOSURE}(I)$ 的定义如下。

① 状态集 I 的 $\varepsilon\_\mathrm{CLOSURE}(I)$ 是一个状态集。

② 状态集 I 的所有状态属于 $\varepsilon\_\mathrm{CLOSURE}(I)$。

③ 若 s 在 I 中，那么从 s 出发经过任意条 ε 弧到达的状态 s'都属于 $\varepsilon\_\mathrm{CLOSURE}(I)$。

从 NFA 转换得到的 DFA 不一定是最简化的，可以通过等价变换将 DFA 进行最小化处理。

4) 正规式与有限自动机之间的转换

(1) 对于Σ上的 NFA M，可以构造一个Σ上的正规式 R，使得 $L(R)=L(M)$。

构造过程分以下两步进行。

① 在 M 的状态转换图中加两个节点 x 和 y。

② 按图 2-1 所示的方法逐步消去 M 中的除 x 和 y 的所有节点。

(2) 对于 Σ 上的每一个正规式 R，可以构造一个 Σ 上的 NFA M，使得 $L(M)=L(R)$。

(3) 构造过程分两步进行。

① 对于正规式 R，可用如图 2-2 所示的拓广状态图表示。

② 通过对正规式 R 进行分裂并加入新的节点，逐步把图转变成每条弧上的标记是 Σ 上的一个字符或ε，转换规则如图 2-3 所示。

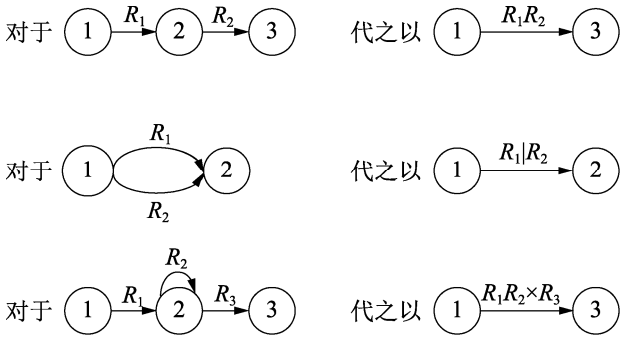


图 2-1　状态转换图(消去中间节点)

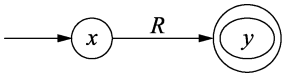


图 2-2　拓广状态图

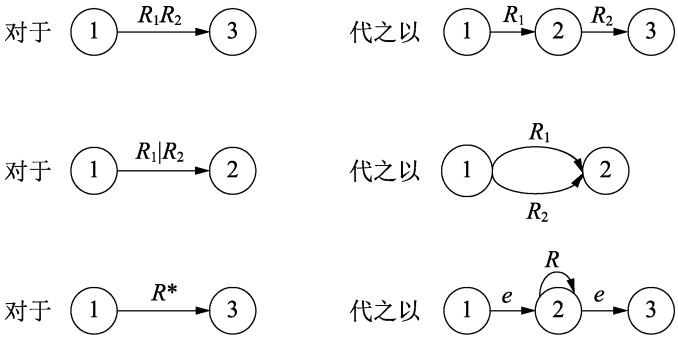


图 2-3　状态转换图(加入新节点)

5)　词法分析器的构造

词法分析器的构造过程如下。

(1)　用正规式描述语言中的单词构成规则。

(2)　为每个正规式构造一个 NFA，用于识别正规式所表示的正规集。

(3)　将构造出的 NFA 转换成等价的 DFA。

(4)　对 DFA 进行最小化处理，使其最简。

(5)　根据 DFA 构造词法分析器。

4．语法分析

语法分析的任务是根据语言的语法规则，分析单词串是否构成短语和句子，同时检查和处理程序中的语法错误。根据产生语法树的方向，语法分析可分为自底向上和自顶向下两类。

自顶向下的分析是对给定的符号串，试图自顶向下地为其构造一棵语法树，或者说从文法的开始符号出发，为其构造一个最佳推导。

自底向上的分析是对给定的符号串，试图自底向上地为其构造一棵语法树，或者说从给定的符号串本身出发，试图将其归约为文法的开始符号。

算符优先文法属于自底向上的分析法，它利用各个算符间的优先关系和结合规则来进

行语法分析，特别是用于分析各种表达式。算符优先文法的任何产生式的右部都会出现两个非终结符相邻的情况，且任何一对终结符之间至多只有 3 种算符关系，即“>”“<”和“=”之一成立。

5．中间代码优化

1) 局部优化

局部优化是在基本块上的优化。基本块是指程序中一个顺序执行的语句序列，其中只有一个入口和一个出口。划分基本块的方法如下。

(1) 求出四元式程序中各个基本块的入口语句。

(2) 对每一入口语句，构造其所属的基本块。它是由该语句到下一入口语句(不包括下一入口语句)，或到一条转移语句(包括该转移语句)，或到一条停语句(包括该停语句)之间的语句序列组成的。

(3) 凡未被纳入某一基本块的语句，都是程序中控制流程无法到达的语句，因而也是不会被执行到的语句，因此可以把它们删除。

一个基本块可以用一个 DAG(有向无环)图表示。在一个基本块内，通常可进行以下 3 种优化，即合并已知量、删除无用赋值和删除多余运算。

2) 控制流图和循环优化

(1) 控制流图。

一个程序的控制流图是一个有向图，其节点是程序中的基本块，它有唯一的首节点，即包含程序第一条语句的基本块，从首节点出发，到控制流图中的每个节点都存在路径。

由程序的各基本块构造相应控制流图的方法是：对于程序中的两个基本块 B_i 和 B_j，若 B_j 紧接着 B_i 被执行，则从 B_i 引一条有向边到 B_j，称 B_i 是 B_j 的直接前驱，B_j 是 B_i 的直接后继。

(2) 循环优化。

循环就是控制流图中具有唯一入口节点的强连通子图，从循环外进入循环时，必须首先经过循环的入口节点。

基于循环的优化处理有代码外提、强度削弱、删除归纳变量等。

6．目标代码生成

代码生成器以经过语义分析或优化后的中间代码为输入，以特定的机器语言或汇编代码为输出。代码生成主要考虑以下问题，即中间代码形式、目标代码形式、寄存器的分配、计算次序的选择。

2.2.1.3 解释程序的基本原理

解释程序是一种语言处理程序，在词法、语法和语义分析方面与编译程序的工作原理基本相同，但在运行用户程序时，它直接执行源程序或源程序的内部形式(中间代码)。因此，解释程序并不产生目标程序，这是它和编译程序的主要区别。

解释程序的结构通常可以分成两部分：第一部分是分析部分，包括通常的词法分析、语法分析和语义分析程序，经语义分析后把源程序翻译成中间代码，中间代码常采用逆波兰表示形式；第二部分是解释部分，用来对第一部分产生的中间代码进行解释执行。

2.2.2　典型例题分析

例 1　C 程序中全局变量的存储空间在 (22) 分配。(2015 年上半年试题 22)

(22) A. 代码区　　B. 静态数据区　　C. 栈区　　D. 堆区

解析：代码区：存放函数体的二进制代码。

栈区：由编译器自动分配释放，存放函数的参数值、局部变量值等。

堆区：一般由程序员分配释放，若程序员不释放，程序结束时可能由操作系统回收。

静态数据区：内存在程序启动的时候才被分配，而且可能直到程序开始执行的时候才被初始化，所分配的内存在程序的整个运行期间都存在，如全局变量、static 变量等。

答案：B

例 2　以下关于语言 $L=\{a^nb^n|n>1\}$的叙述中，正确的是 (48)。(2013 年上半年试题 48)

(48) A. 可用正规式“aa^*bb^*”描述，但不能通过有限自动机识别

B. 可用正规式“a^mb^m”表示，也可用有限自动机识别

C. 不能用正规式表示，但可以用有限自动机识别

D. 不能用正规式表示，也不能通过有限自动机识别

解析：根据正规式和有限自动机的规范，结合题中给出的语言 L，很明显都无法将语言 L 表示和识别出来。

答案：D

例 3　编译过程中，对高级语言程序语句的翻译主要考虑声明语句和可执行语句。对声明语句，主要是将需要的信息正确地填入合理组织的 (49) 中；对可执行语句，则是 (50)。(2013 年上半年试题 49、50)

(49) A. 符号表　　B. 栈　　C. 队列　　D. 树

(50) A. 翻译成机器代码并加以执行　　B. 转换成语法树

C. 翻译成中间代码或目标代码　　D. 转换成有限自动机

解析：编译程序的功能是把用高级语言书写的源程序翻译成与之等价的目标程序(汇编语言或机器语言)。编译程序的工作过程可以分为 6 个阶段，即词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成，实际的编译器中可能会将其中的某些阶段结合在一起进行处理。各个阶段逻辑上可以划分为前端和后端两部分。前端包括从词法分析到中间代码生成各个阶段的工作，后端包括中间代码优化、目标代码生成与优化等阶段。这样，以中间代码为分水岭，把编译器分成了与机器有关的部分和与机器无关的部分。符号表的作用是记录源程序中各个符号的必要信息，以辅助语义的正确性检查和代码生成，在编译过程中需要对符号表进行快速有效地查找、插入、修改和删除等操作。

答案：(49) A　　(50) C

例 4　语言 $L=\{a^mb^n|m\geq 0, n\geq 1\}$的正规表达式是 (21)。(2012 年下半年试题 21)

(21) A．aa^*bb^*　　B．a^*bb^*　　C．aa^*b^*　　D．a^*b^*

解析：$m\geq 0$，a^m 说明可以有 0 个 a 或者多个 a；$n\geq 1$，b^n 说明至少有一个 b 或者多个 b。a^*表示由 0 个或者多个 a 构成的集合，可以表示$\{a^m|\ m\geq 0\}$；b^*表示由 0 个或者多个 b 构成的集合，bb^*才可以表示$\{b^n|\ n\geq 1\}$。

答案：B

例 5 算术表达式“(a-b)*(c+d)”的后缀式是 (21) 。(2014 年下半年试题 21)

(21) A．ab-cd+* B．abcd-*+ C．ab-*cd+ D．ab-c+d*

解析：后缀式是波兰逻辑学家卢卡西维奇发明的一种表达方式，把运算符号写在运算对象的后面，如把 a+b 写成 ab+，这种表示法的优点是根据运算对象和算符的出现次序进行计算，不需要使用括号。

答案：A

例 6 某程序运行时陷入死循环，则可能的原因是程序中存在 (48) 。(2015 年下半年试题 48)

(48) A．词法错误 B．语法错误

C．动态的语义错误 D．静态的语义错误

解析：死循环错误属于典型的语义错误，但静态的语义错误可被编译器发现，到程序真正陷入死循环说明编译器并未发现，所以属于动态语义错误。

答案：C

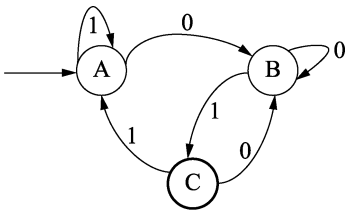
例7 编译程序对高级语言源程序进行编译的过程中，要不断收集、记录和使用源程序中一些相关符号的类型和特征等信息，并将其存入 (22) 中。 (2014年上半年试题22)

(22) A．符号表 B．哈希表 C．动态查找表 D．栈和队列

解析：编译过程中编译程序不断汇集和反复查证出现在源程序中各种名字的属性和特征信息等有关信息。这些信息通常记录在一张或几张符号表中。符号表的每一项有两部分：一部分是名字(标识符)；一部分是名字属性(标识符的有关信息)。编译过程中，每当扫描器(词法分析器)识别出一个名字后，编译程序就查阅符号表，看其是否在符号表中。符号表在编译全过程的地位和作用非常重要，是进行上下文合法性检查和语义处理及代码生成的依据。符号表总体结构的设计和实现是与源语言的复杂性(包括词法结构、语法结构的复杂性)有关，还与对于编译系统在时间效率和空间效率方面的要求有关。

答案：A

例 8 下图所示为一个有限自动机(其中，A 是初态、C 是终态)，该自动机所识别的字符串的特点是 (48) 。(2012 年上半年试题 48)



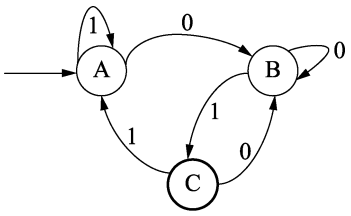
(48) A. 必须以 11 结尾的 0、1 串 B. 必须以 00 结尾的 0、1 串

C. 必须以 01 结尾的 0、1 串 D. 必须以 10 结尾的 0、1 串

解析：有限自动机可识别的字符串，是指从有限自动机的初态出发，存在一条到达终态的路径，其上的标记所构成的字符串。本题 C 是终态，C 的前一状态只能是 B，由 B 到 C 输入的是 1；B 的前一状态可以是 A，也可以是 B，还可以是 C，但输入的都是 0。可见，该有限自动机识别的串必须以 01 结尾。

答案：C

例 9　下图所示为一个有限自动机(其中，A 是初态、C 是终态)，该自动机识别的语言可用正规式 (48) 表示。(2011 年下半年试题 48)



(48) A. (0|1)*01　　B. 1*0*10*1　　C. 1*(0)*01　　D. 1*(0|10)*1*

解析：在正规式中，符号*表示重复若干次(包括 0 次)，符号|表示“或”。在状态 A，可以输入 1 或 0，如果输入 1 还可以回到状态 A，如果输入 0 则直接到达状态 B；在状态 B，可以输入 0 或 1，如果输入 0 则还回到状态 B，而输入 1，则进入状态 C；在状态 C 可以输入 0 或 1，输入 0 到达状态 B，输入 1 到达状态 A，但由于 C 是终态，自动机可识别的语言是由 0、1 构成的字符串的集合，但该集合必须以 01 结束，因此选项 A 正确。

答案：A

例 10　表达式采用逆波兰式表示时，利用 (22) 进行求值。(2015 年下半年试题 22)

(22) A．栈　　B．队列　　C．符号表　　D．散列表

解析：逆波兰将运算符写在操作数之后，因此也称为后缀表达式。将一个普通的中序表达式转换为逆波兰表达式的一般过程是：从左至右扫描表达式，如果当前字符为变量或者为数字，则入栈；如果是运算符，则将栈顶两个元素弹出做相应运算，结果再入栈；以此类推，最后当表达式扫描完，栈里的就是结果。

答案：A

例 11　若一种程序设计语言规定其程序中的数据必须具有类型，则有利于 (22) 。(2011 年上半年试题 22)

① 在翻译程序的过程中为数据合理分配存储单元

② 对参与表达式计算的数据对象进行检查

③ 定义和应用动态数据结构

④ 规定数据对象的取值范围及能够进行的运算

⑤ 对数据进行强制类型转换

(22) A．①②③　　B．①②④　　C．②④⑤　　D．③④⑤

解析：数据是程序操作的对象，类型说明数据占用的内存和存放形式。数据类型不仅可用于在基础机器中完成对值的布局，还可以用于检查表达式中对运算的应用是否正确。

答案：B

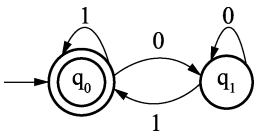
例 12　对高级语言源程序进行编译或解释的过程可以分为多个阶段，解释方式不包含 (48) 阶段。(2015 年上半年试题 48)

(48) A. 词法分析　　B. 语法分析　　C. 语义分析　　D. 目标代码生成

解析：解释方式运行用户程序时，直接执行源程序或者源程序的中间表示形式，不产生源程序的目标程序。

答案：D

例 13 某非确定的有限自动机(NFA)的状态转换图如下图所示(q_0 既是初态也是终态)。以下关于该 NFA 的叙述中，正确的是 (49) 。(2015 年下半年试题 49)



(49) A．其可识别的 0、1 序列的长度为偶数

B．其可识别的 0、1 序列中 0 与 1 的个数相同

C．其可识别的非空 0、1 序列中开头和结尾字符都是 0

D．其可识别的非空 0、1 序列中结尾字符是 1

解析：在初态 q_0 输入 1 又回到 q_0，输入 0 则迁移到状态 q_1，可见在 0 之前可以有任意多个 1(1*)；在 q_1 状态输入 0，则回到 q_1，输入 1 则迁移到终态 q_0，因此在 1 之前可以有任意多个 0(0*)，因此可识别的字符串为 1*00*1。

答案：D

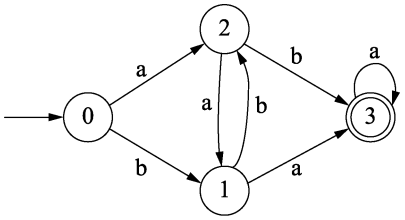
例 14 对高级语言源程序进行编译的过程可以分为多个阶段，分配寄存器的工作在 (48) 阶段进行。(2014 年下半年试题 48)

(48) A．词法分析　　B．语法分析　　C．语义分析　　D．目标代码生成

解析：目标代码生成阶段应考虑直接影响到目标代码速度的 3 个问题：一是如何生成较短的目标代码；二是如何充分利用计算机中的寄存器，减少目标代码访问存储单元的次数；三是如何充分利用计算机指令系统的特点，以提高目标代码的质量。

答案：D

例 15 下图所示的有限自动机中，0 是初始状态，3 是终止状态，该自动机可以识别 (22) 。(2010 年下半年试题 22)

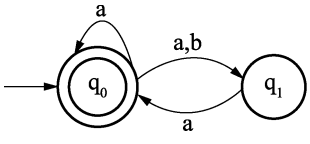


(22) A．abab　　B．aaaa　　C．bbbb　　D．abba

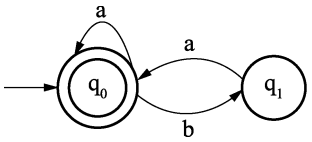
解析：从初始状态到终止状态有多条路径。在状态 0 输入 a 到达状态 2，在状态 2 可输入 a 或 b，输入 a 到达状态 1，输入 b 到达状态 3，状态 3 下输入 a 还回到状态 3；在状态 1 可输入 a 或 b，输入 a 到达状态 3，输入 b 到达状态 2。

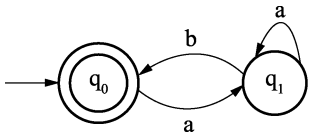
答案：B

例 16 某非确定的有限自动机(NFA)的状态转换图如下图所示(q_0 既是初态也是终态)，与该 NFA 等价的确定的有限自动机(DFA)是 (49) 。(2015 年上半年试题 49)

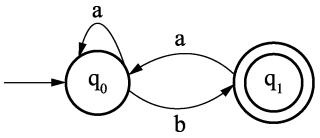


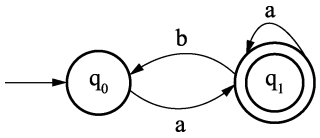
(49) A.　　B.





C.　　D.





解析：一个非确定自动机(NFA)在读入符号串之后，并不确切地知道自动机处于哪个状态。但可以肯定一定处于状态集中的某一状态。该状态集记作$\{q_1,q_2,\cdots,q_k\}$。而一个等价的确定自动机(DFA)读入同样的 w 一定处于某个确定的状态上。这样，都是读入同样的 w，DFA 到达某一个状态，而 NFA 到达某一个状态集。由 w 的任意性，可将 NFA 的所有的状态集和 DFA 的状态一一对应起来。这种对应的前提就是能识别同样的输入串。

答案：A

例 17　以下关于汇编语言的叙述中，错误的是 (50) 。(2010 年下半年试题 50)

(50) A．汇编语言源程序中的指令语句将被翻译成机器代码

B．汇编程序先将源程序中的伪指令翻译成机器代码，然后再翻译指令语句

C．汇编程序以汇编语言源程序为输入，以机器语言表示的目标程序为输出

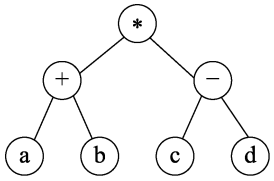
D．汇编语言的指令语句必须具有操作码字段，可以没有操作数字段

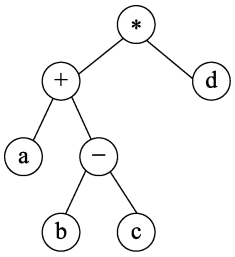
解析：汇编程序的功能是将汇编语言所编写的源程序翻译成机器指令程序。汇编语言源程序语句可分为指令语句、伪指令语句和宏指令语句。指令语句汇编后产生相应的机器代码；伪指令语句指示汇编程序在汇编源程序时完成某些操作，汇编后不产生机器代码。

答案：B

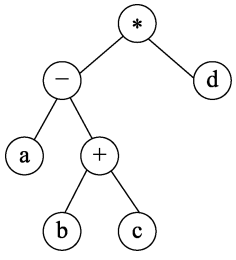
例 18　与算术表达式“(a+(b-c))*d”对应的树是(21)(2015 年上半年试题 21)

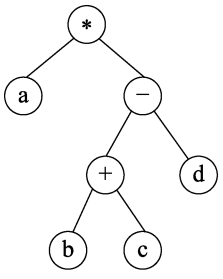
(21) A.　　B.





C.　　D.





解析：表达式用树形来表示时通常采用中缀形式，运算符在树中放在非终端节点的位置上，操作数放在叶子节点处。处理时，首先找到运算级别最低的运算符“ *”作为根节点，继而确定该根节点的左、右子树节点在表达式串中的范围为 a+(b-c)和 d；再在对应的范围内寻找运算级别最低的运算符作为子树的根节点，直到范围内无运算符，则剩余的变量或数为表达式树的叶子。

答案：B

例 19 递归下降分析方法是一种(50)方法。(2015 年上半年试题 50)

(50) A．自底向上的语法分析　　B．自上而下的语法分析

C．自底向上的词法分析　　D. 自上而下的词法分析

解析：在编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程；而在解释方式下，解释程序和源程序要参与到程序的运行过程中，运行程序的控制权在解释程序。解释器翻译源程序时不产生独立的目标程序，而编译器则需要将源程序翻译成独立的目标程序。

答案：A

例20 以下关于实现高级程序设计语言的编译和解释方式的叙述中，正确的是 (48) 。(2014年上半年试题48)

(48) A．在编译方式下产生源程序的目标程序，在解释方式下不产生

B．在解释方式下产生源程序的目标程序，在编译方式下不产生

C．编译和解释方式都产生源程序的目标程序，差别是优化效率不同

D．编译和解释方式都不产生源程序的目标程序，差别在是否优化

解析：在编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程；而在解释方式下，解释程序和源程序要参与到程序的运行过程中，运行程序的控制权在解释程序。解释器翻译源程序时不产生独立的目标程序，而编译器则需要将源程序翻译成独立的目标程序。

答案：A

例 21 对于正规式 0*(10*1)*0*，其正规集中字符串的特点是 (50) 。(2010 年上半年试题 50)

(50) A．开头和结尾必须是 0　　B．1 必须出现偶数次

C．0 不能连续出现　　D．1 不能连续出现

解析：闭包运算符“ *”将其运算对象进行若干次连接，因此 0*表示若干个 0 构成的串，而(10*1)*则表示偶数个 1 构成的串。

答案：B

例 22 将高级语言源程序翻译成机器语言程序的过程，常引入中间代码。以下关于中间代码的叙述中，不正确的是 (22) 。(2014 年下半年试题 22)

(22) A．中间代码不依赖于具体的机器。

B．使用中间代码可提高编译程序的可移植性

C．中间代码可以用树或图表示

D．中间代码可以用栈和队列表示

解析：中间代码是源程序的一种内部表示，或称中间语言。中间代码的作用是可使编

译程序的结构在逻辑上更为简单明确，使用中间代码可提高编译程序的可移植性，常见的有逆波兰记号、四元式、三元式和树。

答案：D

例 23　以下关于编译系统对某高级语言进行翻译的叙述中，错误的是(48)。(2009 年下半年试题 48)

(48) A．词法分析将把源程序看做一个线性字符序列进行分析

B．语法分析阶段可以发现程序中所有的语法错误

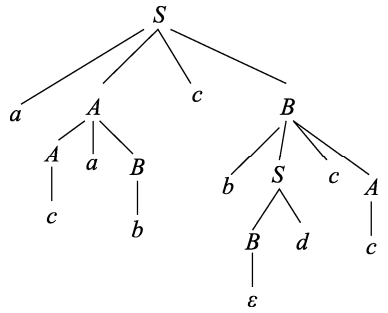
C．语义分析阶段可以发现程序中所有的语义错误

D．目标代码生成阶段的工作与目标机器的体系结构相关

解析：在词法分析阶段，源程序可以简单地被看成一个多行的字符串。这一阶段的任务是对源程序从前到后(从左到右)逐个字符进行扫描，从中识别出一个个“单词”符号，语法分析的任务是在词法分析的基础上，根据语言的语法规则将单词符号序列分解为各类语法单位，检查其中的语法错误；语义分析阶段的主要任务是检查源程序是否包含静态语义错误，并收集类型信息供后面的代码生成阶段使用；目标代码生成是编译器工作的最后一个阶段，这一阶段的任务是把中间代码变化为特定机器上的绝对指令代码、可重定位的指令代码或汇编指令代码，这个阶段的工作与具体的机器密切相关。因此说法 C 中发现所有语义错误是不对的。

答案：C

例 24　由某上下文无关文法 $M[S]$推导出某句子的分析树如下图所示，则错误的叙述是(50)。(2009 年下半年试题 50)



(50) A．该文法推导出的句子必须以 a 开头

B．$acabcbdcc$ 是该文法推导出的一个句子

C．$S \rightarrow aAcB$ 是该文法的一个产生式

D．a、b、c、d 属于该文法的终结符号集

解析：上图所示为某上下文无关文法 $M[S]$推导出某句子的分析树，通过观察上图，只要稍作推导就可推出 $acabcbdcc$ 是该文法推导出的一个句子；看该分析树的第一层分支即可知 $S \rightarrow aAcB$ 是该文法的一个产生式；而 a、b、c、d 因为在图中是分析树的叶子，所以都是该文法的终结符号；右边的 B 分支下有 $S \rightarrow Bd$、$B \rightarrow \varepsilon$，所以该文法推导出的句子不一定是 a 开头的。

答案：A

例 25　以下程序设计语言中，(20) 更适合用来进行动态网页处理。(2014 年上半年试题 20)

(20) A .HTML　　B．LISP　　C. PHP　　D. JAVA/C++

解析：HTML用于处理静态网页；LISP是一种基于λ演算的函数式编程语言。

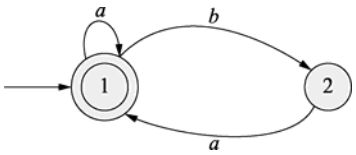
PHP 是一种通用开源脚本语言。语法吸收了 C 语言、Java 和 Perl 的特点，利于学习，使用广泛，主要适用于 Web 开发领域。它可以比 CGI 或者 Perl 更快速地执行动态网页。用 PHP 做出的动态页面与其他的编程语言相比，PHP 是将程序嵌入到 HTML(标准通用标记语言下的一个应用)文档中去执行，执行效率比完全生成 HTML 标记的 CGI 要高许多；PHP 还可以执行编译后代码，编译可以达到加密和优化代码运行的效果，使代码运行更快。

Java 是一种可以撰写跨平台应用软件的面向对象的程序设计语言。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于个人计算机、数据中心、游戏控制台、科学超级计算机、移动电话和互联网。

C++是一个接近系统底层的综合的、支持面向对象和规范编程的程序设计语言，适用于开发要求很高的程序，如大型游戏、大型企业应用、系统应用等。

答案：C

例 26　以下关于下图所示有限自动机的叙述中，不正确的是 (49) 。(2014 年下半年试题 49)



(49) A. 该自动机识别的字符串中 *a* 不能连续出现

B. 自动机识别的字符串中 *b* 不能连续出现

C. 自动机识别的非空字符串必须以 *a* 结尾

D. 自动机识别的字符串可以为空串

解析：图中 *a* 可代表两个步骤：状态 1 -> 1, 状态 2->1。如果两个 *a* 连续出现，则无法区分。

答案：A

例 27　由 *a*、*b* 构造且仅包含偶数个 *a* 的串的集合用正规式表示为 (49) 。(2009 年上半年试题 49)

(49) A．$(a^*a)^*b^*$　　B．$(b^*(ab^*a)^*)^*$　　C.$(a^*(ba^*)^*b)^*$　　D．$(a|b)^*(aa)^*$

解析：本题主要考查考生对闭包概念的理解。

Σ^* 指包括空串 ε 在内的 Σ 上所有字符串的集合。关键在于 Σ^* 可取空串 ε，理解了这个概念就不难看出答案 B 是正确的。

答案：B

例 28　对于大多数通用程序设计语言，用 (50) 描述其语法即可。(2014 年下半年试题 50)

(50) A．正规文法　　B．上下文无关文法

C．上下文有关文法　　D．短语结构文法

解析：上下文无关文法：形式语言理论中一种重要的变换文法，用来描述上下文无关语言，在乔姆斯基分层中称为 2 型文法，由于程序设计语言的语法基本上都是上下文无关

文法，因此应用十分广泛。

答案：B

例29　大多数程序设计语言的语法规则用 (49) 描述即可。(2014年上半年试题49)

(49) A．正规文法　　B．上下文无关文法

C．上下文有关文法　　D．短语结构文法

解析：形式语言理论中一种重要的变换文法，用来描述上下文无关语言，在乔姆斯基分层中称为2型文法。由于程序设计语言的语法基本上都是上下文无关文法，因此应用十分广泛。上下文无关文法拥有足够强的表达力来表示大多数程序设计语言的语法。另外，上下文无关文法又足够简单，使得我们可以构造有效的分析算法来检验一个给定字串是否是由某个上下文无关文法产生的。

答案：B

2.2.3　同步练习

1.　设某上下文无关文法如下：$S\rightarrow 11 \mid 1001 \mid S_0 \mid SS$，则该文法产生的所有二进制字符串都具有的特点是_____。

A．能被 3 整除　　B．0、1 出现的次数相等

C．0 和 1 的出现次数都为偶数　　D．能被 2 整除

2.　编译器对高级语言源程序的处理过程可以划分为词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成等几个阶段，其中，_____并不是每种编译器都必需的。

A．词法分析和语法分析　　B．语义分析和中间代码生成

C．中间代码生成和代码优化　　D．代码优化和目标代码生成

3.　已知某文法 $G[S]$:$S\rightarrow 0S_0$ $S\rightarrow 1$，从 S 推导出的符号串可用_____($n\geqslant 0$)描述。

A．$(010)^n$　　B．0^n10^n　　C．1^n　　D．01^n0

4.　有限自动机(FA)可用于识别高级语言源程序中的记号(单词)，FA 可分为确定的有限自动机(DFA)和不确定的有限自动机(NFA)。若某 DFA D 与某 NFA M 等价，则_____。

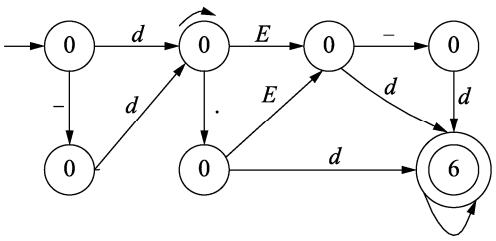
A．DFA D 与 NFA M 的状态数一定相等

B．DFA D 与 NFA M 可识别的记号相同

C．NFA M 能识别的正规集是 DFA D 所能识别的正规集的真子集

D．DFA D 能识别的正规集是 NFA M 所能识别的正规集的真子集

5.　某确定性有限自动机(DFA)的状态转换如下图所示，令 d=0|1|2|…|9，则以下字符串中，能被该 DFA 接收的是_____。



A．3857　　B．1.2E+5　　C．−123.67　　D．0.576E10

6. 属于面向对象、解释型程序设计语言的是_____。(2014 年下半年试题 18)

A．XML　　B．Python　　C．Prolog　　D．C++

7. 集合 $L=\{a^m b^m \mid m \geqslant 0\}$ _____。

A．可用正规式“$a^* b^*$”表示

B．不能用正规式表示，但可用非确定的有限自动机识别

C．可用正规式“$a^m b^m$”表示

D．不能用正规式表示，但可用上下文无关文法表示

8. 编译程序对高级语言源程序进行翻译时，需要在该程序的地址空间中为变量指定地址，这种地址称为_____。

A．逻辑地址　　B．物理地址　　C．接口地址　　D．线性地址

9. _____是指在运行时把过程调用和响应调用所需要执行的代码加以结合。

A．绑定　　B．静态绑定　　C．动态绑定　　D．继承

10. 给定文法 *G*[*S*]及其非终结符 *A*，FIRST(*A*)定义为：从 *A* 出发能推导出的终结符号的集合(*S* 是文法的起始符号，为非终结符)。对于文法 *G*[*S*]：

S→[*L*] | *a*

L→*L*，*S*| *S*

其中，*G*[*S*]包含的 4 个终结符号分别为：*a*，[ ]

则 FIRST(*S*)的成员包括_____。

A．*a*　　B．*a*、[　　C．*a*、[和]　　D．*a*、[、]和，

11. 高级语言源程序的编译过程分若干个阶段，分配寄存器属于_____阶段的工作。

A．词法分析　　B．语法分析　　C．语义分析　　D．代码生成

2.2.4 同步练习参考答案

1．A	2．C	3．B	4．B
5．C	6．B	7．D	8．A
9．C	10．B	11．D	

2.3 本 章 小 结

本章知识点在 2013 年的新大纲中改动不大，只是有一些描述方面的调整。

本章主要要求考生掌握汇编、编译和解释系统的基础知识和基本工作原理，程序设计语言的基本成分，以及各类程序设计语言的主要特点和适用情况。

分析历年的考题可以看出，程序语言基础试题是每年必考的知识点，每年只考 3 道或者 4 道题目。

虽然占的比分不高，但还是需要考生掌握相应的基础知识。考查的重点是形式语言基础、语法和词法分析方法、程序控制结构。复习中还应注意补充程序语言的最新发展方向的知识。

2.4　达标训练题及参考答案

2.4.1　达标训练题

1．文法 $G=(V_T,V_N,P,S)$ 的类型由 G 中的(1)决定。若 G_0=($\{a,b\}$，$\{S,X,Y\}$,P,S)，P 中的产生式及其序号如下。

① $S \rightarrow XaaY$

② $X \rightarrow YY \mid b$

③ $Y \rightarrow XbX \mid a$

则 G_0 为(2)型文法，对应于(3)，由 G_0 推导出句子 *aaaa* 和 *baabbb* 时，所用产生式序号组成的序列分别为(4)和(5)。

(1) A．V_T　　B．V_N　　C．P　　D．S

(2) A．0　　B．1　　C．2　　D．3

(3) A．图灵机　　B．下推自动机　　C．有限状态自动机　　D．其他自动机

(4)～(5) A．13133　　B．12312　　C．12322　　D．12333

2.　编译的优化工作对于下面程序段构造的控制流程图有一个基本块。

```
      A:＝0
      j:=100
      i:=1
loop1: B:=j+1
      C:=B+i
      A:=A+C
      if i=100 goto loop2
      i:=i+1
      goto loop1
loop2: write A
      halt
```

A．1　　B．2　　C．3　　D．4

2.4.2　参考答案

1．(1) C　(2) C　(3) B　(4) D　(5) C

2．D