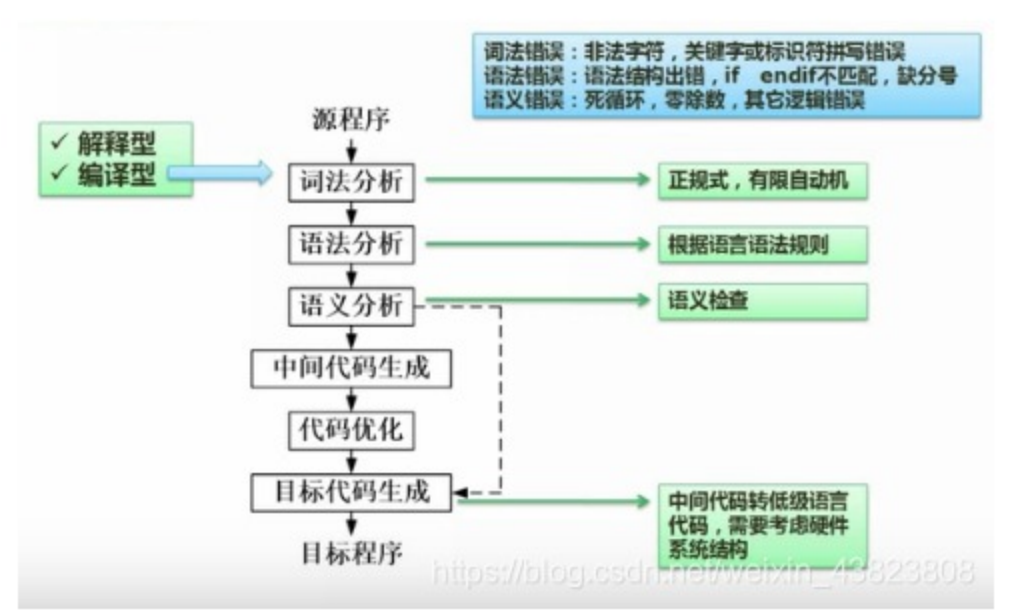
1.主要内容

- 编译与解释
- 文法
- 正规式
- 有限自动机
- 表达式
- 传值与传址
- 多种程序语言特点

2.编译过程



2.1 编译过程概述

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（汇编语言或机器语言）。

2.2 词法分析

词法分析阶段是编译过程的第一个阶段，这个阶段的任务是对源程序从前到后（从左到右）逐个字符地扫描，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，如关键字（或称保留字）、标识符、常数、运算符和分隔符（如标点符号、左右符号）等。词法分析程序输出的"单词"常以二元组的方式输出，即单词种别和单词自身的值。

2.3 语法分析

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在本质上都是对源程序的结构进行分析。

2.4 语义分析

语义分析阶段分析各语法结构的含义，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。
只有语法和语义分析都正确的源程序才能翻译成正确的目标代码。

2.5 中间代码生成

中间代码生成阶段的工作是根据语义分析的输出生成中间代码。"中间代码"是一种简单且含义明确的记号系统，可以有若干种形式，它们的共同特征是**与具体的机器无关。**
最常用的一种中间代码是与汇编语言的指令非常相似的三地址码。

2.6 代码优化

由于编译器将源程序翻译成中间代码的工作是机械的、按固定模式进行的，因此，生成的中间代码往往在时间上和空间上有较大的浪费。当需要生成高效的目标代码时，必须进行优化。
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。

2.7 目标代码生成

目标代码生成阶段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。这个阶段的工作与具体的机器密切相关。

3.文法定义

一个形式文法是一个有序四元组G=(V，T，S，P)，其中：
1）V：非终结符。不是语言组成部分，不是最终结果，可理解为占位符。
2）T：终结符。是语言的组成部分，是最终结果。V∩T=∅
3）S：起始符。是语言的开始符号。
4）P：产生式。**用终结符替代非终结符的规则**。形如α→β

正则闭包：$A^+=A^1\cup A^2\cup A^3\cup\ldots\cup A^n\cup\ldots$（也就是所有幂的组合）。

闭包：$A^*=A^0\cup A^+$（在正则闭包的基础上，加上$A^0=\{\varepsilon\}$）。

例如a^*={a,aa,aaa,…,ε}，而$(ab)^*$={ab,abab,ababab,…,ε}

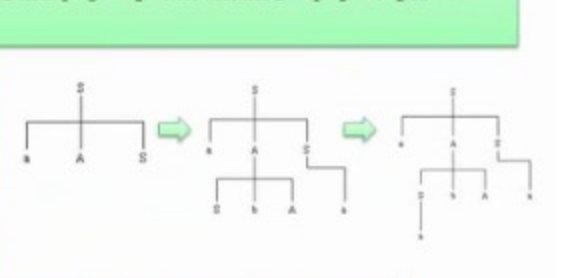
3.1 文法类型

类型	别称	说明	对应自动机
0型	短语文法	G的每条产生式α→β满足a属于V的正则闭包且至少含有一个非终结符，而β属于V的闭包	图灵机
1型	上下文有关文法	G的任何产生式α→β满足|α|<=|β|，仅仅S→ε例外，但S不得出现在任何产生式右部	线性界限自动机
2型	上下文无关文法	G的任何产生式为A→β，A为非终结符，β为V的闭包	非确定的下推自动机
3型	正规文法	G的任何产生式为A→αB或A→α，α属于非终结符的闭包，A、B都属于非终结符	有限自动机

3.2 语法推导树

一棵语法树应具有以下特征：
1. 每个结点都有一个标记，此标记是V的一个符号；
2. 根的标记是S；
3. 若一结点n至少有一个它自己除外的子孙，并且有标记A，则A肯定在V_N中；
4. 如果结点n的直接子孙，从左到右的次序是结点$n_1, n_2, \ldots n_k$，其标记分别是：$A_1, A_2, \cdots, A_k$，那么$A\to A_1, A_2\cdots A_k$，一定是P中的一个产生式。

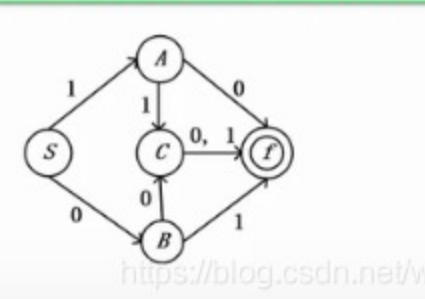
例：
文法G=（{a, b}, {S, A}, S, P），其中：
S→aAS|a;
A→SbA|SS|ba。
请构造句型aabAa的推导树。



S → aAS;　S → a;　A → SbA;　A → SS;　A → ba。

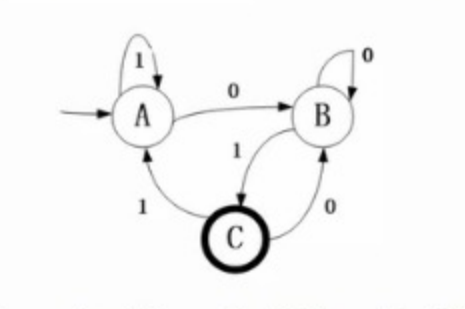
3.3 有限自动机

M=(S,Σ, δ,S0,Z)
1）S是一个有限集，每个元素为一个状态
2）Σ是一个有穷字母表，每个元素为一个输入字符
3）δ是转换函数：是一个单值对照
4）S0,属于S，是其唯一的初态
5）Z是一个终态集（可空）
有限状态自动机可以形象地用状态转换图表示，设有限状态自动机：
DFA = ({S, A, B, C, f}, {1, 0},δ,S, {f}),
其中：
δ(S, 0) = B, δ(S, 1) = A, δ(A, 0) = f, δ(A, 1) = C, δ(B, 0) = C, δ(B, 1) = f,δ(C, 0) = f, δ(C, 1) = f



3.3.1 有限自动机例题

例题：下图所示为一个有限自动机（其中，A是初态、C是终态），该自动机可识别（1）。



（1） A. 0000　B. 1111　C. 0101　D. 1010

这类题相对于正规式的题要简单的多，没有什么技巧，直接按照四个选项给出的表达式去一步一步的推就可以了！！！
由于A是初态，C是终态，我们代入验证四个选项中哪一个可以由初态走到终态，即为正确答案。
A选项：0000，从A开始，由0到达B；再由0，仍然到达B（因为在B这里，0只能是以自身循环），所以0000无法识别。
B选项：1111，从A开始，由1到达A；再由1，仍然到达A（因为在A这里，1只能是以自身循环），所以1111无法识别。
C选项：0101，从A开始，由0到达B；再由1，到达C；再由0，到达B；最后由1，到达终态C，所以0101可以识别。
D选项：1010，从A开始，由1到达A；再由0，到达B；再由1，到达C；最后由0，到达B，所以1010无法识别。

3.4 正规式

正规式是描述程序语言单词的表达式，对于字母Σ，其上的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(ε)={ε}。
② 若a是Σ上的字符，则a是一个正则式，它所表示的正规L(a)={a}。
③ 若正规式r和s分别表示正规集L(r)=L(s)，则
（a）r|s是正规式，表示集合L(r)∪L(s)；
（b）r·s是正规式，表示集合L(r)L(s)；
（c）r*是正规式，表示集合(L(r))*；
（d）(r)是正规式，表示集合L(r)。
仅由有限次地使用上述三个步骤定义的表达式才是Σ上的正规式。由此可见，正规式要么为空，要么由字母、或、连接、闭包运算符组成。其中闭包运算符"*"具有最高的优先级，连接运算具有次高优先级，或运算符"|"具有最低优先级。

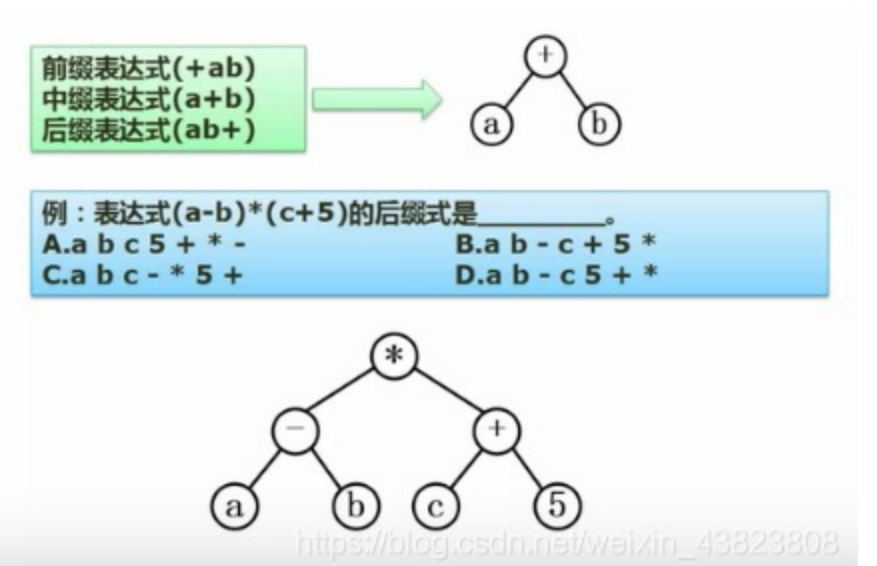
3.4.1 正规式例题

例题：下面文法G[S]它无法识别（1），此文法对应正规式为（2）。
G[S]:
S→aA|bB
A→bS|b
B→aS|a
（1）A. ababab　　B. bababa
C. abbaab　　D. babba
（2）A. (a|b)*　　B.(ab)*
C. (ab|ba)*　　D. (ab)*(ba)*

上面这道例题，其中的 | 表示"或"，*表示重复 [0，+∞）次。那么对于文法G[S]的分析如下：👇👇👇
第一空：👇👇👇
A选项：首先S→aA | bB，可以推出aA；根据A→bS | b，可以推出abS；根据S→aA | bB，可以推出abaA；根据A→bS | b，可以推出ababS；可以发现这样的正规式是一个以ab循环多次的字符串，即可以推出ababab。
B选项：首先S→aA | bB，可以推出bB；根据B→aS | a，可以推出baS；根据S→aA | bB，可以推出babB；根据B→aS | a，可以推出babaS；可以发现这样的正规式是一个以ba循环多次的字符串，即可以推出bababa。
C选项：首先S→aA | bB，可以推出aA；根据A→bS | b，可以推出abS；根据S→aA | bB，可以推出abbB；根据B→aS | a，可以推出abbaS；根据S→aA | bB，可以推出abbaaA；根据A→bS | b，可以推出abbaab。
D选项：首先S→aA | bB，可以推出bB；根据B→aS | a，可以推出baS；根据S→aA | bB，可以推出babB；此时根据B→aS | a，无法推出babb，因为B要么是aS、要么是a，不可能出现b这种情况，所以D选项是错误的！！！

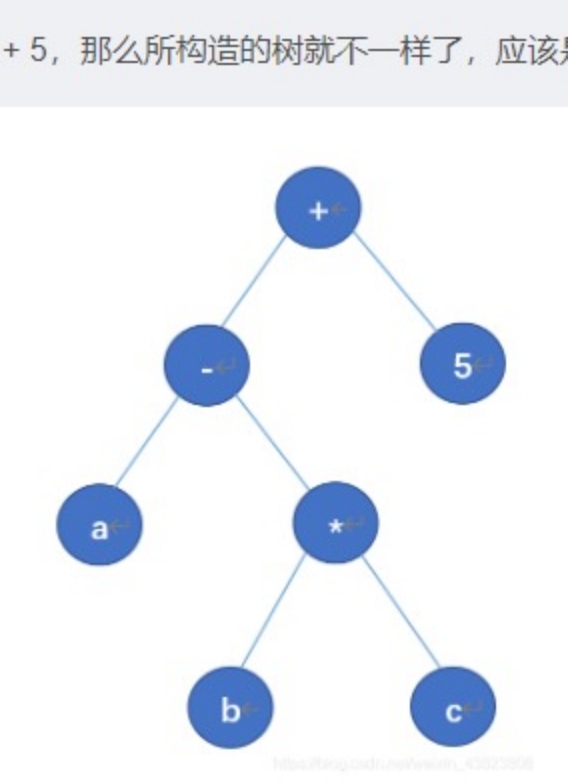
第二空：意思是说这四个选项中哪一个可以将第一空中文法G[S]可以识别的三个选项都表示出来：👇👇👇
A选项：（a | b）*，它可以将由a或b组成的任意串表达出来，也就是这样：a、ab、baa、babba这些都可以。那么它所表达的范围已经超出了文法G[S]可以识别的范围，它无法与文法G[S]保持等价，所以排除A选项。
B选项：（ab）*，它可以将由ab组成的串循环表示 [0，+∞）次，也就是这样：ab、abab、abababab这些都可以。但是它无法表达第一空的B、C两个选项，因为它全部是以ab这样的形式表达的。所以排除B选项。
C选项：（ab | ba）*，它可以生成任意数量的ab串或ba串，也就是这样：ababab、bababa或者abbaab这些都可以，它所表达的范围与文法G[S]完全等价！！！
D选项：（ab）*（ba）*，它的意思是：先来若干个ab串、再来若干个ba串，也就是这样：ababab、bababa、ababbaba这些都可以，但是它无法表示第一空的C选项。所以排除D选项。

4.表达式



将表达式（a-b）*（c+5）构造成树的步骤为：**括号不能出现在树中；按照表达式的计算顺序来依次构造！！！**
第一步肯定是要计算（a-b），之后再计算（c+5），最后将这两者的结果相乘，所构造的树即为上图这种形式：👆👆👆
那么将树构造出来之后，对于前缀、中缀、后缀，无非就是二叉树的的前序、中序、后序遍历的过程：a b - c 5 + *。

如果说这里将表达式中的括号去掉：a - b * c + 5，那么所构造的树就不一样了，应该是下图所示的形式：👇👇👇



5.函数调用——传值与传址

传递方式	主要特点
传值调用	形参取的是实参的值，形参的改变不会导致调用点所传的实参的值发生改变
引用（传址）调用	形参取的是实参的地址，即相当于实参存储单元的地址引用因此其值的改变同时就改变了实参的值

5.1 传值调用

未命名1.cpp

```
#include<stdio.h>
void swap(int x,int y) {
    int t;
    t=x;
    x=y;
    y=t;
    printf("%d %d\n",x,y);
}
int main() {
    int a=3,b=4;
    swap(a,b);
    printf("%d %d\n",a,b);
}
```

E:\计算机专业学习资料和文件\C,C++\未命名1.exe

```
4 3
3 4
--------------------------------
Process exited after 0.3809 seconds with return value 0
请按任意键继续. . .
```

采用传值调用方式时，形参取的是实参的值，修改了形参并不会改变实参的具体值。对于上面这段代码，调用swap函数的语句swap（a，b），进入函数体void swap（int x，int y）的传值调用，虽然函数体内x和y的值进行了交换，第一行打印x和y的值分别为4、3，回到主函数之后，这里因为是传值调用并未改变实参a、b的值，所以a仍然为3，b仍然为4。

5.2 引用调用（传址调用）

未命名1.cpp

```
#include<stdio.h>
void swap(int &x,int &y) {
    int t;
    t=x;
    x=y;
    y=t;
    printf("%d %d\n",x,y);
}
int main() {
    int a=3,b=4;
    swap(a,b);
    printf("%d %d\n",a,b);
}
```

E:\计算机专业学习资料和文件\C,C++\未命名1.exe

```
4 3
4 3
--------------------------------
Process exited after 0.3386 seconds with return value 0
请按任意键继续. . .
```

采用引用调用方式时，形参取的是实参的地址，即修改了形参实际上也就修改了实参（换句话说，形参与实参在这里是一样的）。
对于上面这段代码，调用swap函数的语句swap（a，b），进入函数体void swap（int &x，int &y）的引用调用，函数体内x和y的值进行了交换，第一行打印x和y的值分别为4、3，回到主函数之后，这里因为是引用调用，修改了形参x、y的值，实际上也就修改了实参a、b的值，所以a为4，b为3。

6.各种程序语言特点

1. Fortran语言(科学计算，执行效率高)
2. Pascal语言（为教学而开发的，表达能力强，Delphi）
3. C语言（指针操作能力强，高效）
4. Lisp语言（函数式程序语言，符号处理，人工智能）
5. C++语言（面向对象，高效）
6. Java语言（面向对象，中间代码，跨平台）
7. C#语言（面向对象，中间代码，.Net）
8. Prolog语言（逻辑推理，简洁性，表达能力，数据库和专家系统）

6.1 主要特点

①C：指针操作能力强，高效。
②C++：面向对象，高效。
③C#：面向对象，中间代码，.Net。
④Java：面向对象，中间代码，跨平台。
⑤Python：解释型，面向对象，胶水语言。
⑥Fortran：科学计算，执行效率高。
⑦Pascal：为教学而开发的，表达能力强，Delphi。
⑧Lisp：函数式程序语言，符号处理，人工智能。
⑨Prolog：逻辑推理，简洁性，表达能力，数据库和专家系统。

6.2 语言分类

①编译型：逐段编译，生成可执行程序exe等，执行效率高。
②解释型：逐句，解释器，跨平台，执行效率低。

7.数据类型与程序控制结构