编译器设计专题实验
实验报告

实验 7 拉链回填

班级：计算机 2102
姓名：李芝塬
学号：2216113163

目录

实验要求

实现拉链回填

实验分析

拉链回填需要在四元式的 result 部分中，先留空，待计算出结果后，再返回填写 result 的值。

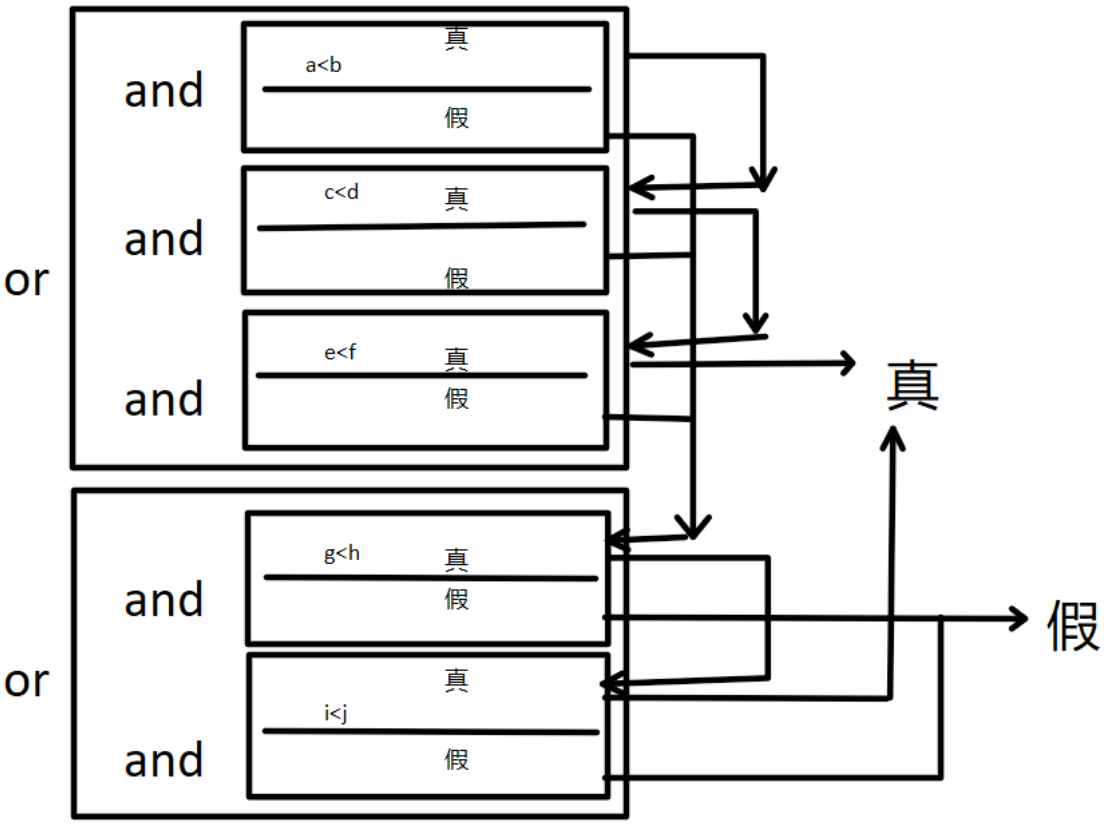
为了简化难度，本次实验只完成对逻辑运算的拉链回填，即：

输入一个逻辑表达式，输出对应的四元式组。

对于一个逻辑表达式，计算顺序是先 AND 再 OR，所以使用 OR 将整个表达式划分为大块，每个 AND 表达式为一个小块。对于表达式:

```
a< b and c < d and e < f or g < h and i < j
```

划分方式如下：



每次进行一次逻辑运算，为真则跳到下一个 AND 块，为假则直接跳到下一个 OR 块。如果 OR 块中最后一个 AND 也是真，则这个 OR 块为真，整个逻辑表达式可以直接判断为真；如果最后一个 OR 块中某一个 AND 为假，则直接为假。每个 AND 块占 2 条指令，每个 OR 块的指令数取决于 AND 的数量。

实验结果

本程序实现了对逻辑表达式的拉链回填。

每次读取一行，即一个完整的逻辑表达式。为了简化程序，要求所有符号之间使用空格划分，没有空格的字符串视为一个 token。

```
#include <bits/stdc++.h>
using namespace std;
vector<string> symbols;
int main()
{
    int yes = 1, nextOR = 100, lineNum = 100;
    string a;
    cout << "所有符号和名词之间使用空格分开！" << endl;
    while (1)
    {
        cout << "请输入逻辑运算表达式(输入q退出程序):" << endl;
        getline(cin, a);
        if (a == "q")
            break;
        a += " #";          // 加个终止符号
        stringstream ss(a); // a赋值给ss,
        string s;
        while (ss >> s) // ss 以空格间隔输出字符串 并赋值给s
        {
            if (s == "or" || s == "#")…
            else if (s == "and")…
            else…
        }
        yes = 1, nextOR = 100, lineNum = 100; // 重置
        a = "";
    }
}
```

每次获取一个字符 s，如果是 OR 或 AND 或终结符，就分别进行处理，如果不是就保存到数组中。数组中只保存一个 OR 块，当遇到 OR 或者终结符就需要分析然后清空数组。

如果是 OR，就需要分析这个 OR 块中的所有表达式。每个表达式有 3 个字符串，例如 a<b 就有"a","<","b"这 3 个字符串。AND 不占用字符串，OR 也不占用，所以每次从 symbols 数组中读取 3 个字符串为一个表达式。如果这个表达式为真，则继续下一个 AND 块，即地址加 2；如果表达式为假，则跳转到下一个 OR 块（使用 nextOR 保存下一个 OR 块的位置）。

如果是遇到终结符，表明没有下一个 OR 块了，设置 nextOR 为 0，表示下一个跳转到 0。

在 for 循环遍历中，n-3 至 i-1 是当前 OR 块中最后一个 AND 块的内容，这个 AND 块需要特殊处理，因此不放在 for 中；i<n-3 是前面的 AND 块，循环处理，如果计算为真则跳转到 2 行之后，如果为假则跳转到下一个 OR 块，即 nextOR。

对于最后一个 AND 块，如果也是真，则这个 OR 块为真，不需要再进行后面的运算，直接判断整个表达式为真，否则跳转到下一个 OR 块。

OR 块相关代码如下：

```
if (s == "OR" || s == "#")
{
    if (s == "OR")
        nextOR += 2;
    else
```

```
        nextOR = 0;
    int n = symbols.size();
    fOR (int i = 0; i < n - 3; i += 3) // 每个逻辑运算一定占3个位置
    {
        printf("%d(j%s,%s,%s,%d)\n", lineNum, symbols[i + 1].c_str(),
symbols[i].c_str(), symbols[i + 2].c_str(), lineNum + 2); // AND 真--通
过 lineNum 继续
        lineNum++;
        printf("%d(j,_,_,%d)\n", lineNum, nextOR);
        nextOR = lineNum++;
    }
    printf("%d(j%s,%s,%s,%d)\n", lineNum, symbols[n - 2].c_str(),
symbols[n - 3].c_str(), symbols[n - 1].c_str(), yes); // OR\end 情况 真--
跳转
    yes = lineNum++;
    printf("%d(j,_,_,%d)\n", lineNum, nextOR);
    lineNum++;
    symbols.clear();
}
```

如果读取到 AND，表明读取完了一个表达式。对于每一次比大小，为真占一条指令，为假占用一条指令，所以需要占用 2 条指令，则下一个 OR 块的位置会增加 2,。因此在读到 AND 时，只需要将 nextOR 加 2。

如果读到其他字符，表明这是运算的一部分，保存起来即可。

```
else if (s == "AND")
{
    // tt 中不保存 AND
    nextOR += 2; // 存一下，去 OR 的情况下处理再一起处理
}
else
{
    symbols.push_back(s);
}
```

运行结果如下：

```
lzy@iZj6c34iaw7zzdcsox1suvZ:~/compile/7$ ./run.sh
编译实验7
当前时间
Thu Jul  4 07:38:47 PM CST 2024
开始编译
输入内容
a < b and c > d or e == f
a < b and c > d or e == f and g != h and i < j
Tom == Jerry and Mary != Lily and John == Mike and Kate == Lisa
a <= b or c < d and e >= f and g != h and i == j and k == l and m == n and o == p and
q >= r and s <= t and u < v and w >= x and y == z
q
开始运行
```

```
输出内容
所有符号和名词之间使用空格分开!
请输入逻辑运算表达式(输入q退出程序):
100(j<,a,b,102)
101(j,_,_,104)
102(j>,c,d,1)
103(j,_,_,101)
104(j==,e,f,102)
105(j,_,_,0)
请输入逻辑运算表达式(输入q退出程序):
100(j<,a,b,102)
101(j,_,_,104)
102(j>,c,d,1)
103(j,_,_,101)
104(j==,e,f,106)
105(j,_,_,0)
106(j!=,g,h,108)
107(j,_,_,105)
108(j<,i,j,102)
109(j,_,_,107)
请输入逻辑运算表达式(输入q退出程序):
100(j==,Tom,Jerry,102)
101(j,_,_,0)
102(j!=,Mary,Lily,104)
103(j,_,_,101)
104(j==,John,Mike,106)
105(j,_,_,103)
106(j==,Kate,Lisa,1)
107(j,_,_,105)
```

```
请输入逻辑运算表达式(输入q退出程序):
100(j<=,a,b,1)
101(j,_,_,102)
102(j<,c,d,104)
103(j,_,_,0)
104(j>=,e,f,106)
105(j,_,_,103)
106(j!=,g,h,108)
107(j,_,_,105)
108(j==,i,j,110)
109(j,_,_,107)
110(j==,k,l,112)
111(j,_,_,109)
112(j==,m,n,114)
113(j,_,_,111)
114(j==,o,p,116)
115(j,_,_,113)
116(j>=,q,r,118)
117(j,_,_,115)
118(j<=,s,t,120)
119(j,_,_,117)
120(j<,u,v,122)
121(j,_,_,119)
122(j>=,w,x,124)
123(j,_,_,121)
124(j==,y,z,100)
125(j,_,_,123)
请输入逻辑运算表达式(输入q退出程序):
```

在本程序中，如果判定整个表达式为真，则跳转到 1；如果判断整个表达式为假，则跳转到 0。在实际程序中，应该分别跳转到对应的代码块中，所以需要先把代码块的长度算出来，再回填这里的 0 和 1。

实验总结

这次实验本来需要与前面几次所做的语法分析、语义分析相结合，再结合符号表的内容综合完成本次实验。但是由于难度太大，所以在此只完成了对逻辑表达式跳转的四元式的回填。平心而论，本次实验完成得不太好，一方面是没有完成课件上所要求的内容，另一方面，为了完成实验也是偷工减料，读取输入时的要求很高，以逃避对词法分析的要求。整个实验的内容量也很小，只是完成了拉链回填中对 if-else 语句的一部分内容。实在是羞愧。

到现在为止，这学期的编译实验课也算结束了。这门课对学生要求很宽松，给分也合理，整体难度并不高，对动手能力的提升很大，比以前的大部分实验课都有用。但是也有不合理之处。这个实验课的各个部分并不连贯，词法分析给的 token 表并不符合 C 语言标准，语法分析的文法只是简单的算术表达式，离完成 C 语言编译还差十万八千里，但是在实验 7 中突然要求写出 MIPS 汇编代码，写出栈帧，写出调用序列等，难度陡增。各个部分之间没有联系，没有衔接，在完成后面的内容时不能引用前面的实验结果，令人摸不着头脑。

另外，或许是大部分实验要求已经在上课时讲过，所有 PPT 上的内容比较精简。如果没有在课堂上认真记录实验要求，课后很难看懂实验 PPT。这一点在实验 7 上尤其明显。所以还是需要学生上课时认真听讲。

总体而言，这门课相当不错，瑕不掩瑜。对于我自己的评价是前期比较认真，后期有些松懈，但是所有实验都是我自己独立完成，没有抄袭同学。确实有参考网上的资料和 AI，但是绝不是简单的复制粘贴，而是经过自己的研究后再进化，再改写。

这是我本科期间最后一门实验课，祝老师工作顺利，祝同学们学习进步，更上一层楼！

附录

源代码：Github
参考链接：腾讯云开发者社区