## 切换导航 Farseerfc的小窝

文

繁體

<u>简体</u>

**English** 

日本語

- <u>▲ About</u>
- <u> Links</u>
- ≜ About
- <u>Links</u>
- □ Import
- □ Life
- <u>□ Tech</u>

- □<u>Import</u>
- <u>Life</u>
- <u>Tech</u>
- Q
  - 搜索
- 搜索
- <u>児档</u>
- 9

C++ Tricks 3.2 标号、goto,以及switch的 实现 <u>迢</u>

## 3.2 标号、goto,以及switch的实现

goto语句及标号(label)是最古老的C语言特性,也是最早被人们抛弃的语言特性之一。像汇编语言中的jmp指令一样,goto语句可以跳转到同一函数体中任何标号位置:

```
void f()
{int i=0;
Loop: //A label
++i;
if(i<10)goto Loop; //Jump to the label
}</pre>
```

在原始而和谐的早期Fortran和Basic时代,我们没

有if then else,没有for和while,甚至没有函数的概念,一切控制结构都靠goto(带条件的或无条件的)构件。软件工程师将这样的代码称作"意大利面条"代码。实践证明这样的代码极容易造成混乱。

自从证明了结构化的程序可以做意大利面条做到的任何事情,人们就开始不遗余力地推广结构化设计思想,将goto像猛兽一般囚禁在牢笼,标号也因此消失。

标号唯一散发余热的地方,是在switch中控制分支 流程。

很多人不甚了解switch存在的意义,认为它只是大型嵌套if then else结构的缩略形式,并且比if语句多了很多"不合理"的限制。如果你了解到switch在编译器内部的实现机制,就不难理解强加在switch之上的诸多限制,比如case后只能跟一个编译期整型常量,比如用break结束每一个case。首先看一个switch实例:

switch (shape.getAngle())

```
case 3: cout<<"Triangle";break;
case 4: cout<<"Square";break;
case 0:case1: cout<<"Not a sharp!";break;</pre>
default: cout<<"Polygon";
}
仟何程序员都可以写出与之对应的if结构:
int i= getAngle(shape);
if (i==3) cout<<"Triangle";
else if(i==4) cout<<"Square";
else if(i==0||i==1) cout<<"Not a sharp!";
else cout<<"Polygon";
看起来这两段代码在语义上是完全一样的,不是
么?
不!或许代码的执行结果完全一样,但是就执行效
率而言,switch版本的更快!
```

要了解为什么switch的更快,我们需要知道编译器 是怎样生成switch的实现代码的: 首先,保留switch之后由{}括起来的语具体,仅将其 中case、default和break替换为真正的标号: switch (getAngle(shape)) { \_case\_3: cout<<"Triangle";goto \_break; \_case\_4: cout<<"Square"; goto \_break;</pre> \_case\_0:\_case\_1: cout<<"Not a sharp!"; goto break; default: cout<<"Polygon"; break: } 随后,对于所有出现在case之后的常量,列出一张 只有goto的跳转表,其顺序按case后的常量排列:

goto case 0;

```
goto _case_1;
goto case 3;
goto _case_4;
然后、计算case之后的常量与跳转表地址之间的关
系,如有需要,在跳转表中插入空缺的项目:
100105: goto _case_0;
100110: goto case 1;
100115: goto _default; //因为没有case 2,所以插
入此项以条转到default
100120: goto _case_3;
100125: goto _case_4;
假设一个goto语句占用5个字节,那么在本例中,
goto的地址=case后的常量*5+100105
之后,生成跳转代码,在其余条件下跳转至
```

之后,生成跳转代码,在其余条件下跳转至 default,在已知范围内按照公式跳转,全部的实现 如下:

```
{
int i= getAngle(shape);
if (i<0||i>=5)goto _default;
i=i*5+100105; //按照得出的公式算出跳转地址
goto i; //伪代码,C中不允许跳转到整数,但是汇编
允许
100105: goto case 0;
100110: goto case 1;
100115: goto default;
100120: goto case 3;
100125: goto case 4;
_case_3: cout<<"Triangle";goto _break;
_case_4: cout<<"Square"; goto _break;
_case_0:_case_1: cout<<"Not a sharp!"; goto
break;
```

```
_default: cout<<"Polygon";
break:
```

}

经过这样处理整个switch结构,使得无论switch后的变量为何值,都可以通过最多两次跳转到达目标代码。相比之下if版本的代码则采用线性的比较和跳转,在case语句很多的情况下效率极低。

由此,我们也可以知道,为什么case后跟的一定是编译期整型常数,因为编译器需要根据这个值制作跳转表。我们可以明白为什么case与case之间应该用break分隔,因为编译器不改变switch语句体的结构,case其本身只是一个具有语义的标号而已,要想跳出switch,就必须用break语句。