C＋＋ワンポイントレッスン

C++と++Cの違いについて

ソフトウエア研究会

前置きと後置きの演算子

- Ｃ＋＋には、Cと同様の変数をインクリメント・デクリメントする演算子（＋＋、－－）がある。

- Cの時は、おもに整数,ポインタ変数などに対し、アセンブラ命令のインクリメント・デクリメントに相当する命令を直接記述することにより高速で効率の良いコード生成を可能としてきた。

- Ｃ＋＋では、オブジェクトの演算子をユーザーが定義できるようになったため、演算子の持つ特性と役割が大きく拡張された。

前置き＋＋と後置き＋＋の違い
レジスタ変数時

- Cの時と同様に、レジスタ変数における前置きと後置きのコード生成を考える。

C++での記述

```
if (i++ < 10) Hoge(i)
```

生成されるアセンブラ

```
mov ebx, eax
inc eax
cmp ebx, 10
jnc L1
push eax
call Hoge
L1:
```

C++での記述

```
if (++i < 11) Hoge(i);
```

生成されるアセンブラ

```
inc eax
cmp eax, 11
jnc L1
push eax
call Hoge
L1:
```

前置き＋＋と後置き＋＋の違い
レジスタ変数時　結果

前置き＋＋の場合のデメリット

- 命令が1つされる。（メモリの増加、速度の低下）
- レジスタ変数が1つ消費される。(最適化の効率低下）
- コンパイル時間の増加。（最適化の余地が多いため、最適化ルーチンのコストが増加する）

※実際には、この程度のコードならほとんどのコンパイラは最適化が可能なので、実行時の差異は生じません。

※生成されるアセンブラはイメージです。実際にはコンパイラによって生成されるコードは異なります。

前置き＋＋と後置き＋＋の違い
クラスオブジェクト時

クラスオブジェクトでオーバーライドされた演算子における前置きと後置きのコード生成を考える。

C++での記述

```
CFoo foo;
if (foo++ < 10) Hoge(foo)
```

インライン展開されるコード

```
CFoo tmp = foo;
foo.m_value.plus1();
if (operator < (tmp.m_value, 10))
{
  Hoge(foo)
}
```

C++での記述

```
CFoo foo;
if (++foo < 11) Hoge(foo);
```

インライン展開されるコード

```
if (operator < (foo.m_value.plus1() , 11))
{
  Hoge(foo)
}
```

前置き＋＋と後置き＋＋の違い
クラスオブジェクト時

CFooの実装は以下のようになる。

```
struct CValue {
  CValue & plus1();
}
bool operator < (const CValue&, int);

struct CFoo {
  CValue m_value;
  CFoo() {}
  bool operator < (int n) { return m_value  < n; }
  CFoo& operator ++(int) {
    m_value .plus1();
    return *this;
  }
  CFoo operator ++() {
    CFoo tmp = *this;
    m_value .plus1();
    return tmp;
  }
}
```

前置き＋＋と後置き＋＋の違い
クラスオブジェクト時の結果

前置き＋＋時のデメリット

- CFooのオブジェクトの生成の追加
- CFooオブジェクトの代入（コピー）の追加（インライン展開されない場合は2回発生する）
- コード量の増加
- 最適化コストの増加

前置き＋＋と後置き＋＋のパフォーマンス測定

オブジェクト	map iterator++	++ map iterator	vector iterator ++	++ vector iterator
実行時間（リリースビルド）	1.300815	0.300459	0.567246	0.566541
比率	100.00%	23.10%	43.61%	43.55%
実行時間（デバッグビルド）	1.361722	0.108110	1.203101	0.157290
比率	100.00%	7.94%	88.35%	11.55%

測定環境　Athlon64 3.0GHz x2 VisualStudio 2009