編譯器期末報告
10 個有關最佳化的例子
B083040029 邱品諺

1. Elimination of Common Sub-expression

事先計算共同的 sub-expression 並存入變數，避免重複計算。右邊的程式碼為左邊經過最佳化的程式碼。

$$X := a + (b * c);$$
$$Y := D + (b * c);$$

$$Temp := (b * c);$$
$$X := a + Temp;$$
$$Y := D + Temp;$$

2. Compile Time Computation

常數的運算先行計算。右邊的程式碼為左邊經過最佳化的程式碼。

$$A = (2 * 3) + B \qquad A = 6 + B$$

3. Boolean Expression Optimization

```
If C1 or C2 Then S1
```

```
If C1 Then S1
Else If C2 Then S1
```

4. Loop Optimization

把 while 迴圈每次都需要計算的運算式先計算好並存入變數，並將不需要的變數初始化移出迴圈。右邊的程式碼為左邊經過最佳化的程式碼。

```
Bound := 10;
While (I <= Bound-2) do
 While (I <= 10) do
 Begin
      X := 1;
      Y := X + Z;
 End;
```

```
Bound := 10;
t := Bound-2;
While (I <= t) do
 X := 1;
 While (I <= 10) do
 Begin
      Y := X + Z;
 End;
```

5. 迴圈中的邏輯順序變換

將原本在迴圈內，且與迴圈的執行結果無關的判斷式移至迴圈外。由於迴圈內的 if 判斷式不會受到迴圈內容影響，故經由最佳化後可從左邊的程式碼變為右邊的程式碼。

```
for i in range(1,5):
    if a == b:
      print i
```

```
if a == b:
    for i in range(1,5):
        print i
```

6. 迴圈融合

若相鄰的兩個迴圈跑的次數相同且相依，合併兩迴圈可減少迴圈控制需要的時間。經過最佳化可由上面的兩個迴圈簡化為下面的單個迴圈。

```
for i in range(1,n_data):
    phones.append(data[i].phone)

for j in range(1,n_data):
    addresses.append(data[j].address)
```

```
for i in range(1,n_data):
    phones.append(data[i].phone)
    addresses.append(data[i].address)
```

7. Branch If Optimization

判斷條件、方式相同，合併成一個 if statement 執行，經最佳化後可變為右邊的程式碼。

```
void f (int *p) {
    if (p) g(1);
    if (p) g(2);
    return;
}
```

```
void f (int *p)  {
    if (p) {
        g(1);
        g(2);
    }
    return;
}
```

8. Value Range Optimization

因為變數 i 必然是大於等於 1 的正整數，if statement 必會通過，因此可刪除 if 條件判斷。下方為上方程式碼最佳化後的程式碼。

```
for (int i = 1; i < 100; i++) {
    if (i) g();
}

for (int i = 1; i < 100; i++) {
    g();
}
```

9. Integer mod Optimization

在大部分的硬體上，除法指令需要較多的 CPU Cycle，因此可考慮利用功能等價的指令來取代。下方為上方程式碼最佳化後的程式碼。

```
int f (int x, int y) {
    return x % y;
}
```

```
int f (int x, int y) {
    int tmp = x & (y - 1);
    return (x < 0) ? ((tmp == 0) ? 0 : (tmp | ~(y - 1))) : tmp;
}
```

10. Access pattern Optimization

存取特定 Class 或是 Structure 的成員時，可事先快取，以減少記憶體操作的次數。下方為上方程式碼最佳化後的程式碼。

```
for (i = 0; i < 10; i++)
    arr[i] = obj.i + volatile_var;
```

```
t = obj.i;
for (i = 0; i < 10; i++)
    arr[i] = t + volatile_var;
```