Nguyên lý và phương pháp lập trình

Tối ưu hóa vòng lặp và logic

TS. Nguyễn Tuấn Đăng

Nội dung

- Các biến đổi vòng lặp
  - Chuyển các phát biểu ra khỏi vòng lặp
  - Giảm các kiểm tra điều kiện
    - Các phần tử cầm canh
  - Loại bỏ vòng lặp
  - Kết hợp các vòng lặp

Nội dung

- Các biến đổi logic
  - Sử dụng các biểu thức tương đương
  - Ngưng kiểm tra điều kiện khi đã biết kết quả
  - Thứ tự kiểm tra các điều kiện
  - Tính toán trước các hàm

1. Các biến đổi vòng lặp

– Chuyển các phát biểu ra khỏi vòng lặp

– Giảm các kiểm tra điều kiện

  + Các phần tử cầm canh

– Giải phóng vòng lặp

– Kết hợp các vòng lặp

Chuyển các phát biểu ra khỏi vòng lặp

- Ý tưởng: Nếu có một biểu thức hay một khối phát biểu cho kết quả không đổi trong vòng lặp thì chuyển nó ra ngoài vòng lặp
- Loại bỏ việc tính toán lại một biểu thức nhiều lần (cho ra cùng kết quả).
- Ví dụ 1:

```
for (int x = 1; x < n; x++)
{
        p(x) = rate * cost(x) * inflator;
 }
```

Chuyển các phát biểu ra khỏi vòng lặp

trở thành

```
inflatedRate = rate * inflator;
for (int x = 1; x < n; x++)
{
    p(x) = inflatedRate * cost(x);
}
```

- Chú ý là phải phải gom các thành phần thích hợp để chuyển ra ngoài vòng lặp.

Chuyển các phát biểu ra khỏi vòng lặp

- Ví dụ 2:

```
for (int i = 1; i < k; i++) {
        if (a < b) {
                p(i);
        } else {
                q(i);
        }
}
```

Chuyển các phát biểu ra khỏi vòng lặp

trở thành

```
if (a < b) {
  for (int i=1; i<k; i++) { p(i); }
} else {
  for (int i=1; i<k; i++) { q(i); }
}
```

- Hai vòng lặp được duy trì song song

Giảm các kiểm tra điều kiện

- Ý tưởng: Nên có càng ít các kiểm tra điều kiện vòng lặp càng tốt, tốt nhất chỉ nên có một kiểm tra điều kiện vòng lặp.

Các phần tử cầm canh

- Ý tưởng: Nếu vòng lặp là một mảng tìm kiếm một chiều thì đặt một phần tử vào cuối dãy cần tìm kiếm và loại bỏ điều kiện so sánh kiểm tra trong chỉ số vòng lặp
- Ví dụ: Tìm trong một phần tử trong đoạn (1..n)

```
i = 1;
while (item(i) != searchValue && i < n) {
        i = i + 1;
}
if (item(i) == searchValue) { ... }
```

Các phần tử cầm canh

trở thành

```
i = 1;
save = item(n+1);
item(n+1) = searchValue;
while (item(i) != searchValue) {
  i = i+1;
}
item(n+1) = save;
if (i <= n) { ... }
```

Các phần tử cầm canh

- Thông thường ít khi cần ghi nhớ và phục hồi lại phần tử ở vị trí cầm canh

Loại bỏ vòng lặp

- Ý tưởng: Loại bỏ (một phần) chi phí tính toán các chỉ số vòng lặp bằng cách xây dựng một biểu thức với các thành phần xác định
- Ví dụ:

```
sum = 0;
for (i = 1; i == 5, i++) {
  sum = sum + x(i);
}
```

trở thành

```
sum = x(1) + x (2) + x(3) + x(4) + x(5);
```

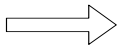
Loại bỏ vòng lặp

- Khi chặn trên không biết, có một cách tiếp cận là:

  (1) Giải phóng k bước lặp đầu tiên, tách k bản sao của phần thân với kiểm tra điều kiện,

  (2) Đơn giản hóa kết quả bằng cách áp dụng các biến đổi phụ

```
for (i=1; i==10; i++) {
     a(i) = i * i;
}
```



```
a(1) = 1;
a(2) = 4;
...
a(10) = 100;
```

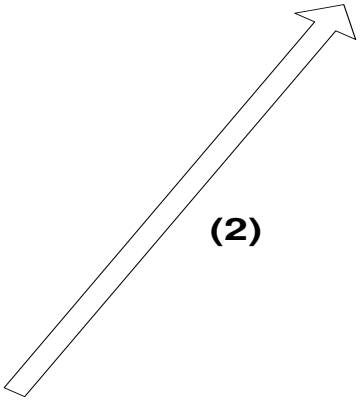
General approach:

(1) unroll 1st k loops

(2) simplify

```
i = 1;
while (i <= n) {
     a(i) = i * i;
     i = i + 1;
}
```

(1)



(2)

```
i = 1;
if (i < n) return;
a(i) = i * i;
i = i + 1;
if (i > n) return;
a(i) = i * i;
i = i + 1;
while (i <= n) {
     a(i) = i * i;
     i = i + 1;
}
```

```
if (1 > n) return;
a(1) = 1;
if (2 > n) return;
a(2) = 4;
i = 3;
while (i <= n) {
     a(i) = i * i;
     i = i + 1;
}
```

(2')

```
switch (n) {
  case 1 :
    a(1) = 1;
    break;
  case 2 :
    a(1) = 1;
    a(2) = 4;
    break;
  default :
    if (1 > n ) {
      a(1) = 1;
      a(2) = 4;
      i = 3;
      while (i <= n) {
        a(i) = i * i;
        i = i + 1;
      }
    }
  }
```

Kết hợp các vòng lặp

- Ý tưởng: Nếu hai vòng lặp thao tác trên cùng các đối tượng thì kết hợp hai thân vòng lặp vào cùng một vòng lặp
- Ví dụ:

```
for (i = 1; i < k; i++) {
    [Body A];
}
for (i = 1; i < k; i++) {
    [Body B];
}
```

Kết hợp các vòng lặp

trở thành

```
for (i = 1; i < k; i++) {
    [Body A];
    [Body B];
}
```

- Lưu ý rằng các phát biểu của Body A và Body B được đan xen vào nhau.

Kết hợp các vòng lặp

- Xem biến đổi sau:

```
public void p(a,b,n) {
    for (i = 1; i < n; i++) {
        a(i) = a(i + 1);
    }
    for (i = 1; i < n; i++) {
        b(i) = b(i + 1);
    }
}
```

Kết hợp các vòng lặp

trở thành

```
public void p(a, b, n) {
    for (i = 1; i < k; i++) {
        a(i) = a(i + 1);
        b(i) = b(i + 1);
    }
}
```

2. Các biến đổi logic

- Sử dụng các biểu thức tương đương
- Ngưng kiểm tra điều kiện khi đã biết kết quả
- Thứ tự kiểm tra các điều kiện
- Tính toán trước các hàm

Sử dụng các biểu thức tương đương

- Ý tưởng: Nếu việc đánh giá một biểu thức logic quá phức tạp, thay thế nó bằng một biểu thức tương đương nhưng đơn giản hơn
- Ví dụ 1:

```
if (square(x) > 0) {
    […]
} else {
    […]
}
```

Sử dụng các biểu thức tương đương

trở thành

```
if (x !=0) { // square(x)>0 iff x !=0
    […]
 } else {
    […]
}
```

Sử dụng các biểu thức tương đương

- Ví dụ 2: (Giả sử có các số thực dương)

if (sqrt(x) < sqrt(y)) {...}

trở thành

if (x < y) {...} // sqrt(x) < sqrt(y) iff x < y

Ngưng kiểm tra điều kiện nếu biết được kết quả

- Ý tưởng: Không cần kiểm tra thêm điều kiện nếu không cần thiết
- Ví dụ 1:

```
if (p(x) && b(x)) { ... }
```

trở thành

```
if (p(x)) {
        if (b(x)) {
        }
}
```

Ngưng kiểm tra điều kiện nếu biết được kết quả

Ví dụ 2:

```
if (p(x) || b(x)) { ... }
```

trở thành

```
if (p(x)) { ... } // ngưng if p(x)=true
else if  (b(x)) { ... }
```

Ngưng kiểm tra điều kiện nếu biết được kết quả

- Ví dụ 3:

```
sum = 0;
for    (j = i; j < n; j++) {
           sum = sum + x(j);
}
if      (sum> cutoff) { ... }
```

Ngưng kiểm tra điều kiện nếu biết được kết quả

Trở thành

```
sum = 0;
j = i;
while (j <= n && sum <= cutoff) {
    sum = sum + x( j);
    j = j + 1;
}
if (sum > cutoff) { ... }
```

Thứ tự kiểm tra các điều kiện

- Ý tưởng: Các kiểm tra logic có thể được sắp xếp sao cho các kiểm tra có chi phí thấp và thường xuyên đúng nằm ở trước các kiểm tra có chi phí cao và ít khi đúng
- Ví dụ: Giả sử các có các vị từ **a** và **b**, chi phí kiểm tra **a** 100mSec, và **b** là 1mSec.

```
if (a && b) {
    p(x);  [cost=101mSec * …]
} else {
    p(y);  [cost=101mSec * …]
}
```

Thứ tự kiểm tra các điều kiện

```
if (a) {
    if (b) {
        p(x); [101mSec * …]
    } else {
    p(y); [101mSec * …]
    }
} else {
        p(y); [100mSec * …]
}
```

Thứ tự kiểm tra các điều kiện

```
if (b) {
    if (a) {
        p(x); [101mSec * …]
    } else {
        p(y); [101mSec * …]
    }
} else {
        p(y); [1mSec * …]
}
```

Tính toán trước các biểu thức

- Ý tưởng: Thay thế bằng một bảng các giá trị tính toán trước cho các biểu thức phức tạp (trên một miền giá trị hữu hạn)
- Ví dụ:

```
Type characterType = (UpperCase,
LowerCase, …);
var cType: characterType;
case inputChar of
        A..Z: cType := UpperCase;
        a..z:  cType := LowerCase;
```

Tính toán trước các biểu thức

trở thành

```
var cType: characterType;
    cTypeTable: array[char] of characterType;
    cType := cTypeTable[inputChar];
```