애플리케이션 최적화

한국기술교육대학교
온라인평생교육원

학습내용

- Optimization
- 최적화 작업

학습목표

- 최적화의 의미와 목적에 대해 설명할 수 있다.
- 최적화가 필요한 코드를 찾을 수 있고 필요한 최적화 작업을 수행할 수 있다.

Optimization

1 최적화 개요

최적화란?

- 정보공학에서 시스템을 수정하여 어떠한 면의 작업을 더 효과적으로, 또는 자원을 덜 사용하도록 만드는 작업

컴퓨터 프로그램은 더 빠르게 실행되거나
기억 장치 또는 자원을 덜 차지하게 하여 운영하도록 개선

유지보수의 편의성을 고려한 코드 개선

안정성 확보를 위한 최적화

2 최적화 기법

성능 체크 → 최적화 대상 선정

코드 분석 → 최적화 향상

Optimization

2 최적화 기법

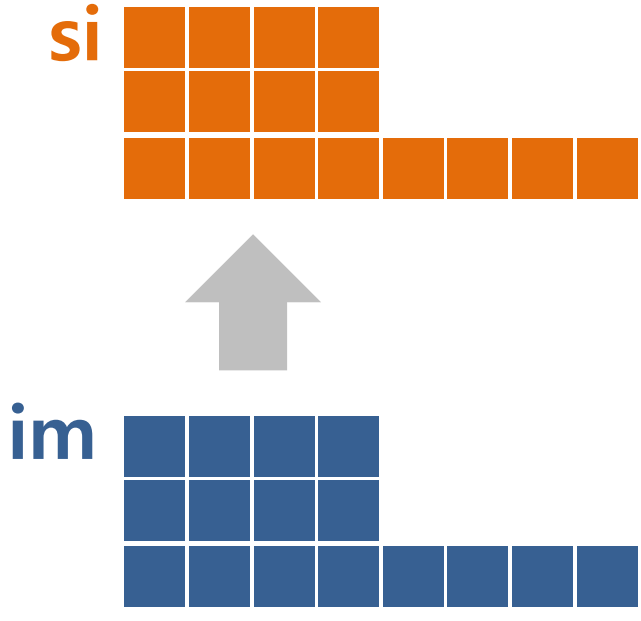
1. 병목지점 찾기(소요시간)
2. 구조체 복사 시 대입연산자 대신 라이브러리 함수를 이용
3. 구조체 전달은 포인터를 이용
4. 함수의 매개변수를 축소
   - 4개 이하 : 레지스터 이용
   - 4개 이상 : 스택 이용
5. 4바이트 이상 전달 시 포인터를 이용
6. 4개이상인경우인자를구조체로선언하고구조체포인터를매개변수로전달
7. const를 적절히 활용
8. 2의 n 제곱을 곱하는 연산은 쉬프트 연산 수행
9. 실수연산을 축소
10. 소수점 이하 2자리까지만 필요한 연산
    - 실수의 연산에서 100을 곱한 후 연산을 해 100으로 나눔
11. 지역변수를 최대한 활용
12. 전역변수 사용을 최소화

최적화 작업하기

1 매개변수 최적화

1 구조체는 포인터로 전달

```
struct test {
   int a,b;
   double c;
};
void sub( struct test si )
{
     ......
}
int main()
{
   struct test im;
    ......
    ......
    sub( im );
    ......
}
```



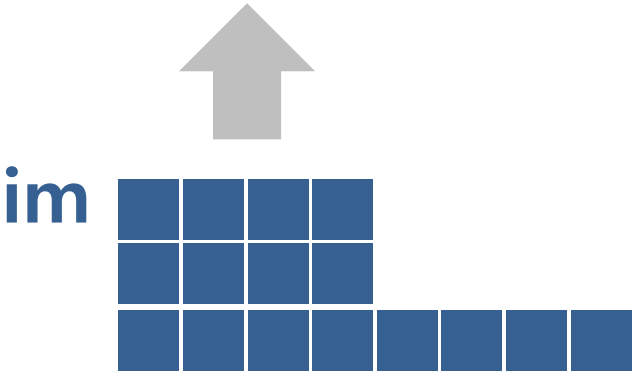
최적화 작업하기

1 매개변수 최적화

1 구조체는 포인터로 전달

```
struct test {
   int a,b;
   double c;
};
void sub( struct test *si )
{
     ......
}
int main()
{
   struct test im;
    ......

    ......
    sub( &im );
    ......
}
```



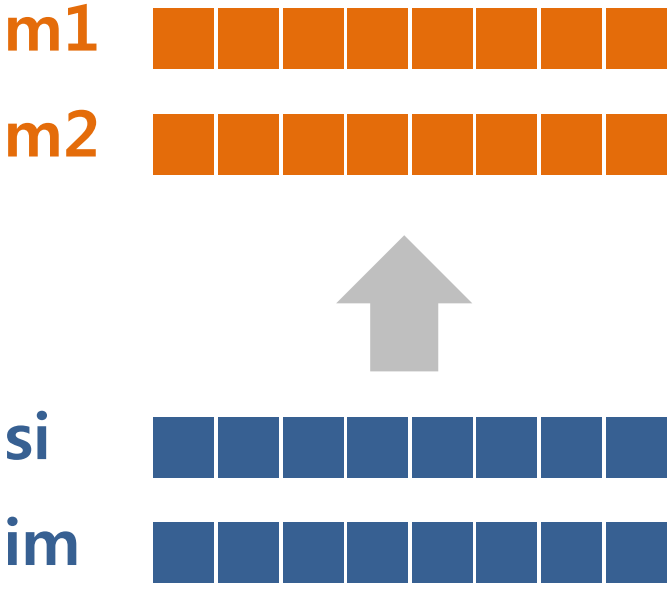
최적화 작업하기

1 매개변수 최적화

2 바이트(Byte) 이상의 데이터는 포인터로 전달

```
void sub( double m1, double m2 )
{
    ……
}
int main()
{
   double im, si ;
    ……

    ……
    sub( im, si );
    ……
}
```



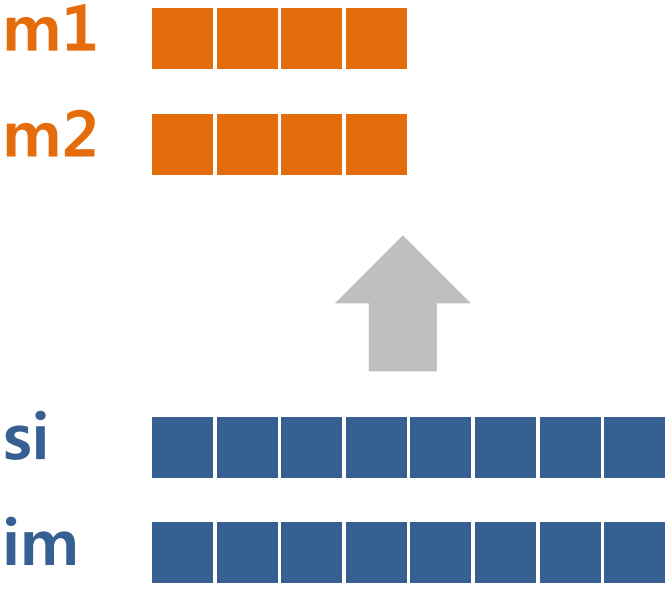
최적화 작업하기

1 매개변수 최적화

2 바이트(Byte) 이상의 데이터는 포인터로 전달

```
void sub( double *m1, double m2 )
{
    ......
}
int main()
{
   double im, si ;
    ......

    ......
    sub( &im, &si );
    ......
}
```

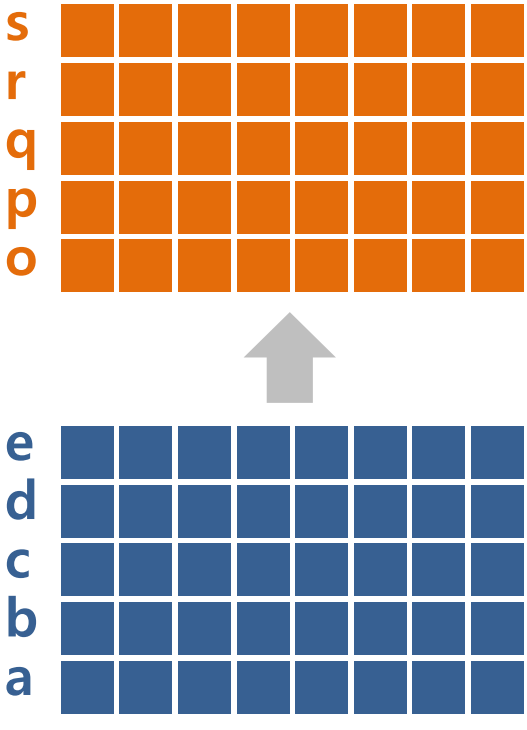


최적화 작업하기

1 매개변수 최적화

3 여러 개의 매개변수를 구조체로 전달

```
struct test {
   int a,b;
   double c;
};
void sub( double o, double p, double q, double r, double s )
{
    ......
}
int main()
{
   double a,b,c,d,e;
   struct test im;
    ......
    ......
    sub( a,b,c,d,e);
    ......
}
```

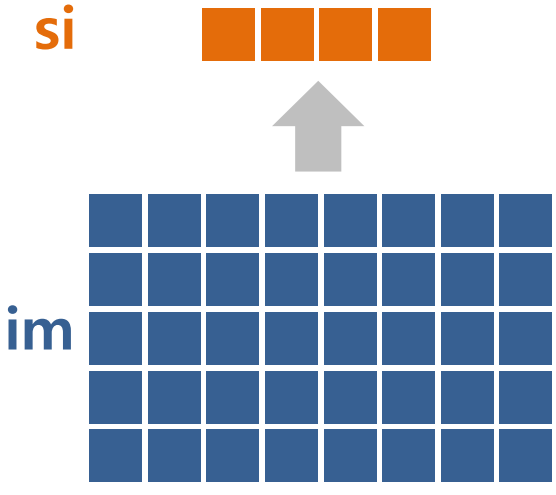


최적화 작업하기

1 매개변수 최적화

3 여러 개의 매개변수를 구조체로 전달

```
struct test {
   double a,b,c,d,e;
};
void sub( struct test *si )
{
     ......
}
int main()
{
   double a,b,c,d,e;
   struct test im;
    ......
   im.a = a;
    ......
    sub( &im );
    ......
}
```



최적화 작업하기

2 연산의 최적화

1 실수 연산을 최소화(정수의 연산으로)

```
int main()
{
   double a=5.0, b=3.0, c;
   double d=15.34, e=3.0, f;
   c = a * b;

   f = d * e;

}
int main()
{
   int p,q,r;
   double a=5.0, b=3.0, c;
   double d=15.34, e=3.0, f;
   p = a;
   q= b;
   r = p * q;

   p = d * 100;              //소수점 이하 2자리의 배정도
   q = e * 100;
   f = ( p * q )/100;
}
```

최적화 작업하기

2 연산의 최적화

2 변수 스코프 고려하기

```
int a, b, c, d;
void sub()
{
     ......
}

int main()
{
   int a;

}
```

```
int main()
{
   int i;

   while( ){
      i = 6;
   }


}
```

```
int main()
{


   while( ){
      int i;
      i = 6;
   }


}
```

최적화 작업하기

3 안정성 확보를 통한 최적화

1 const 활용

```
void sub( const double m1, double m2 )
{
    m1 = 5.9;
    m2 = 4.7
}
int main()
{
   double im, si ;
    ……

    ……
    sub( im, si );
    ……
}
```

학습정리

1. Optimization

- 최적화란 프로그램이 최소의 자원 사용으로 최소 시간에 수행될 수 있도록 코드를 개선하는 과정을 말함
- 유지보수의 편의성을 고려한 코드 개선도 포함됨
- 안정성 확보도 코드 최적화 기법 중 하나에 해당됨

2. 최적화 작업하기

- 실수연산보다 정수연산이 더 빠름
- 4바이트 이상의 매개변수는 포인터로 전달함
- 여러 개의 동일한 데이터 형의 매개변수는 배열로 전달함
- 여러 개의 다른 데이터 형의 매개변수는 구조체로 전달함