



고급프로그래밍

병렬프로그래밍

Professor Jeong, Mun-Ho

Robot Vision & Intelligence Laboratory
Kwangwoon University
(02-940-5625, mhjeong@kw.ac.kr)

Schedule

주차	주제		과제	퀴즈
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++	C++ 프로그래밍의 기본 (3), 클래스와 객체 (4)	1	1
3		객체생성과 사용 (5)	2	2
4		함수와 참조 (6, 3/26), 복사 생성자와 함수중복(7)	3	3
5		static, friend, 연산자중복 (8, 4/2), 연산자중복 상속(9)	4	4
6		상속 (10, 4/9), 가상함수와 추상클래스 (11)		5
7		템플릿과 STL (12, 4/16), 입출력(13)	5	
8		중간고사		
9		파일 입출력(14), 예외처리 및 C 사용(15)		6
10		람다식(16, 5/7) , 멀티스레딩(17, 5/9)	6	7
11		멀티스레딩(18, 5/14), 고급문법(19, 5/16)		8
12		고급문법 2(20, 5/21), 고급문법 3(21, 5/23)		
13	병렬프로그래밍	병렬프로그래밍(22, 5/28)	7	
14		병렬프로그래밍(23, 6/4), 퀴즈(6/7), 병렬프로그래밍(24, 6/8, 비대면)		9
15		기말고사		

Today's Topics

- 보조지시어

보조지시어(Clauses)

- 데이터 공유 속성 지정
- default
- shared
- private
- firstprivate
- lastprivate
- reduction

default

- 병렬영역에서 변수가 자동으로 `shared` 되는 것을 방지
- `default(shared)` : 이렇게 지정할 필요 없음
- `default(none)` : 모든 변수는 `private` 혹은 `shared`로 따로 지정해야함

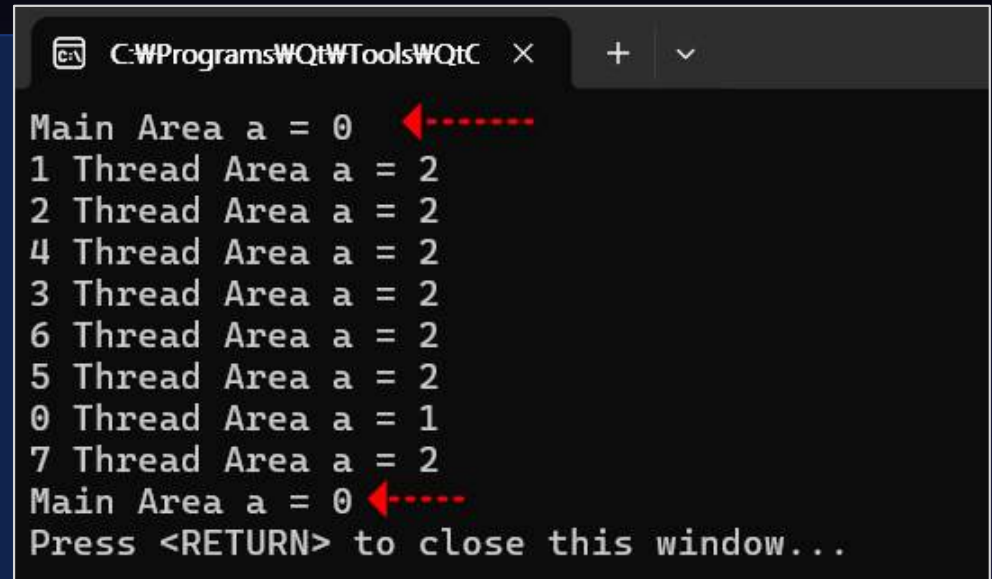
default

```
#include <iostream>
#include <omp.h>
#include <math.h>
#include <chrono>
#include <thread>

int main()
{
    int a=0;
    printf("Main Area a = %d\n", a);

    #pragma omp parallel default(none) private(a)
    {
        if(omp_get_thread_num() ==0) a = 1;
        else
            a = 2;

        printf("%d Thread Area a = %d\n", omp_get_thread_num(), a);
    }
    printf("Main Area a = %d\n", a);
}
```



```
C:\Programs\Qt\Tools\QtC
Main Area a = 0
1 Thread Area a = 2
2 Thread Area a = 2
4 Thread Area a = 2
3 Thread Area a = 2
6 Thread Area a = 2
5 Thread Area a = 2
0 Thread Area a = 1
7 Thread Area a = 2
Main Area a = 0
Press <RETURN> to close this window...
```

default

```
#include <iostream>
#include <omp.h>
#include <math.h>
#include <chrono>
#include <thread>

int main()
{
    int a=0;
    printf("Main Area a = %d\n", a);

    #pragma omp parallel default(none) //private(a)
    {
        if(omp_get_thread_num() ==0) a = 1;
        else a = 2;

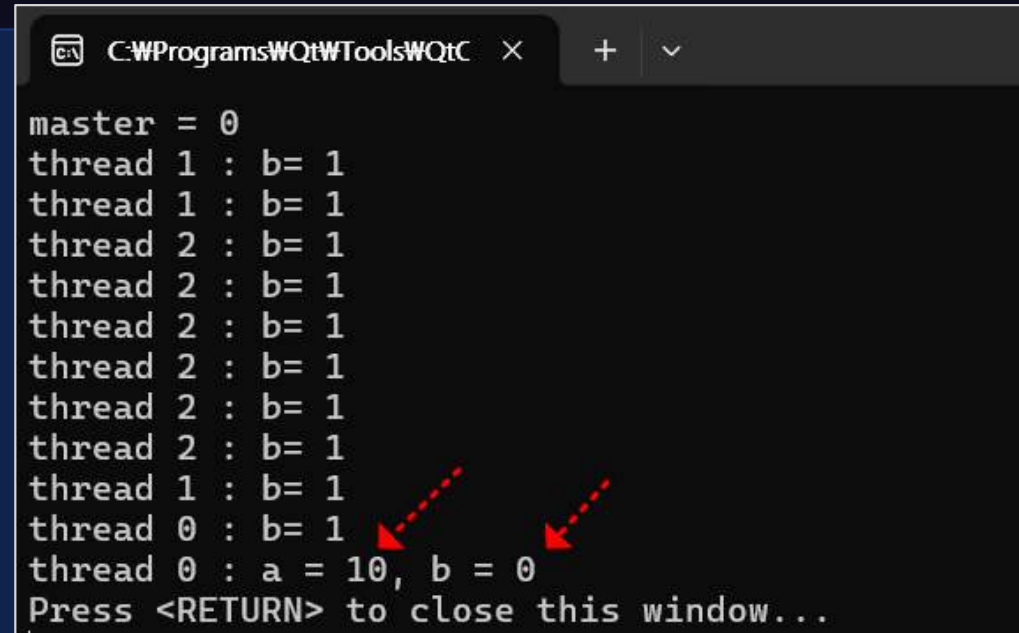
        printf("%d Thread Area a = %d\n", omp_get_thread_num(), a);
    }
    printf("Main Area a = %d\n", a);
}
```

❌	variable 'a' must have explicitly specified data sharing attributes
	explicit data sharing attribute requested here
❗	'a' not specified in enclosing 'parallel'

shared

- 지정된 변수를 모든 스레드가 공유함

```
int main()
{
    int a=0, b=0;
    int master = omp_get_thread_num();
    printf("master = %d\n", master);
    #pragma omp parallel num_threads(3)
    {
        #pragma omp single
        for(int i=0; i<10; i++)
        {
            #pragma omp task shared(a) firstprivate(b)
            {
                #pragma omp atomic ← - - - -
                a++;
                #pragma omp atomic ← - - - -
                b++;
                printf("thread %d : b= %d\n", omp_get_thread_num(), b);
            }
        }
        #pragma omp single
        printf("thread %d : a = %d, b = %d\n", omp_get_thread_num(), a,b);
    }
}
```



```
C:\Programs\Qt\Tools\QtC x + v
master = 0
thread 1 : b= 1
thread 1 : b= 1
thread 2 : b= 1
thread 2 : b= 1
thread 2 : b= 1
thread 2 : b= 1
thread 2 : b= 1
thread 2 : b= 1
thread 1 : b= 1
thread 0 : b= 1
thread 0 : a = 10, b = 0
Press <RETURN> to close this window...
```


private

- 레퍼런스 변수, const 변수는 private으로 사용할 수 없다 - 컴파일 에러

```
11 int main()
12 {
13     float a = 0;
14     float &Ra = a;
15
16
17
18 #pragma omp parallel private(Ra)
19 {
20     Ra = 1;
21 }
22
23 printf("Ra = %.1f\n", Ra);
24
25
26 return 0;
27 }
28
```

private

- 지정된 변수를 스레드 로컬 메모리 영역에서 생성함
- 데이터 값은 전달되지 않음

```
int main()
{
    float A[5] = { 1,2,3,4,5 };
    printf("Main Memory  %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0],A[1],A[2],A[3],A[4]);

    #pragma omp parallel private(A)
    {
        A[0] = 7;
        A[1] = 7;

        printf("Thread Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0],A[1],A[2],A[3],A[4]);
    }

    printf("Main Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0],A[1],A[2],A[3],A[4]);
}
```

private

- 지정된 변수를 스레드 로컬 메모리 영역에서 생성함
- 데이터 값은 전달되지 않음

```
int main()
{
    float A[5] = { 1,2,3,4,5 };
    printf("Main Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0], A[1], A[2], A[3], A[4]);

    #pragma omp parallel private(A)
    {
        A[0] = 7;
        A[1] = 7;

        printf("Thread Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0], A[1], A[2], A[3], A[4]);
    }

    printf("Main Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0], A[1], A[2], A[3], A[4]);
}
```

```
C:\Programs\Qt\Tools\QtC x + v
Main Memory 1.0, 2.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 0.0, 0.0, -561865.5
Thread Memory 7.0, 7.0, 0.0, 0.0, -433001.8
Thread Memory 7.0, 7.0, 0.0, 0.0, -561787.5
Thread Memory 7.0, 7.0, 0.0, 0.0, -561896.5
Thread Memory 7.0, 7.0, 0.0, 0.0, 0.0
Thread Memory 7.0, 7.0, 0.0, 0.0, -561990.5
Thread Memory 7.0, 7.0, 0.0, 0.0, -562068.5
Thread Memory 7.0, 7.0, 0.0, 0.0, -433094.8
Main Memory 1.0, 2.0, 3.0, 4.0, 5.0
Press <RETURN> to close this window...
```

firstprivate

- private와 동일하나 데이터 전달됨(초기화)

```
int main()
{
    float A[5] = { 1,2,3,4,5 };
    printf("Main Memory  %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0],A[1],A[2],A[3],A[4]);

    #pragma omp parallel firstprivate(A)
    {
        A[0] = 7;
        A[1] = 7;

        printf("Thread Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0],A[1],A[2],A[3],A[4]);
    }

    printf("Main Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0],A[1],A[2],A[3],A[4]);
}
```

firstprivate

- private와 동일하나 데이터 전달됨(초기화)

```
int main()
{
    float A[5] = { 1,2,3,4,5 };
    printf("Main Memory  %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0], A[1], A[2], A[3], A[4]);

    #pragma omp parallel firstprivate(A)
    {
        A[0] = 7;
        A[1] = 7;

        printf("Thread Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0], A[1], A[2], A[3], A[4]);
    }

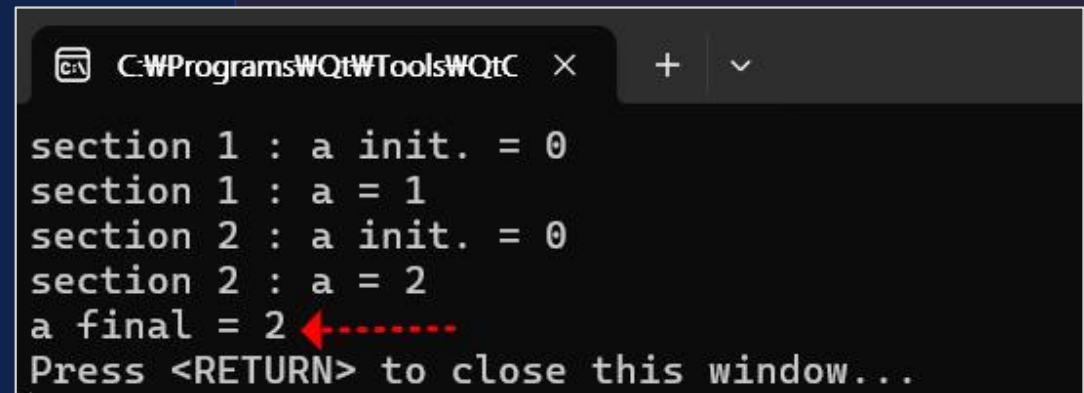
    printf("Main Memory %.1f, %.1f, %.1f, %.1f, %.1f\n", A[0], A[1], A[2], A[3], A[4]);
}
```

```
C:\WPrograms\Qt\Tools\QtC  x  +  v
Main Memory  1.0, 2.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Thread Memory 7.0, 7.0, 3.0, 4.0, 5.0
Main Memory 1.0, 2.0, 3.0, 4.0, 5.0
Press <RETURN> to close this window...
```

lastprivate

- `private`와 동일하나 메인 영역에 결과값을 전달가능

```
int main()
{
    int a = 0;
    #pragma omp parallel
    {
        #pragma omp sections firstprivate(a) lastprivate(a)
        {
            #pragma omp section
            {
                printf("section 1 : a init. = %d\n", a);
                a = 1;
                printf("section 1 : a = %d\n", a);
            }
            #pragma omp section
            {
                printf("section 2 : a init. = %d\n", a);
                a = 2;
                printf("section 2 : a = %d\n", a);
            }
        }
        printf("a final = %d\n", a);
    }
}
```



C:\WPrograms\Qt\Tools\QtC x + v

```
section 1 : a init. = 0
section 1 : a = 1
section 2 : a init. = 0
section 2 : a = 2
a final = 2 ←-----
Press <RETURN> to close this window...
```

reduction

- `private` 동일하나 메인 영역으로 전달 가능하고
- 지정된 연산자에 따라 결과를 취합하여 전달함
- 초기화
 - 복수개 가능 : `(+:a,b)`

오퍼레이터	변수 초깃값
+	0
*	1
-	0
&	~0
	0
^	0
&&	1
	0

reduction

- 각 스레드별로 `sum = 0`로 초기화
- 각 스레드의 `sum`의 총합을 마스터 스레드에 전달함

```
11 int main()
12 {
13     int sum = 0, i = 0;
14     int Data[1000] = { 0, };
15
16     for(i=0; i<1000; i++)
17         Data[i] = i + 1;
18
19     #pragma omp parallel for reduction(+:sum)
20     for(i=0; i<1000; i++)
21         sum += Data[i];
22
23     printf("Total = %d\n", sum);
24
25     return 0;
26 }
```


reduction

