315 Programlama Dilleri

Yrd. Doç. Dr. Ahmet Arif AYDIN

Object-Oriented Programming

- Nesne-tabanlı programlama kavramı ilk defa SIMULA 67 (1960) ile ortaya çıkmıştır.
- 1980 yılında saf nesne tabanlı Smalltalk 80 ortaya çıkmıştır.

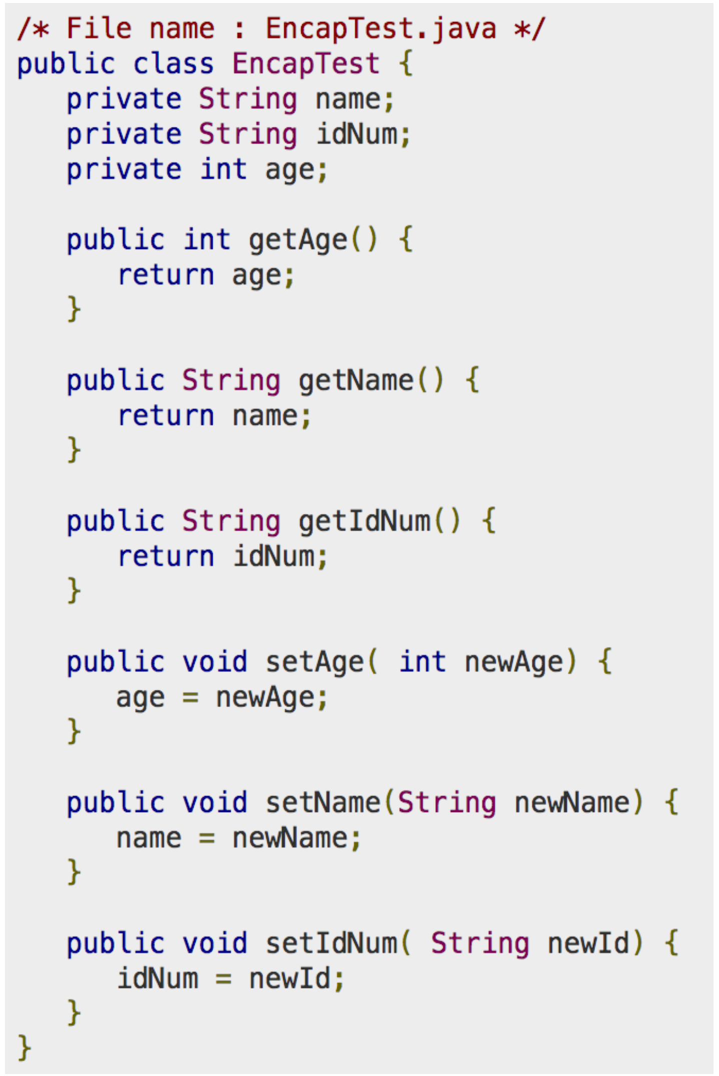
- Nesne tabanlı (object-oriented) programlama dillerinin üç temel özelliği bulunmaktadır:
  - Abstract Data Types
  - Inheritance
  - Dynamic Binding

Object-Oriented Programming

Abstract Data Types (Soyut veri tipleri)

- Encapsulation (kapsulleme)
- İnformation hiding (bilgi gizleme)

Object-Oriented Programming



```java
/* File name : EncapTest.java */
public class EncapTest {
   private String name;
   private String idNum;
   private int age;

   public int getAge() {
      return age;
   }

   public String getName() {
      return name;
   }

   public String getIdNum() {
      return idNum;
   }

   public void setAge( int newAge) {
      age = newAge;
   }

   public void setName(String newName) {
      name = newName;
   }

   public void setIdNum( String newId) {
      idNum = newId;
   }
}
```

Encapsulation Information hiding

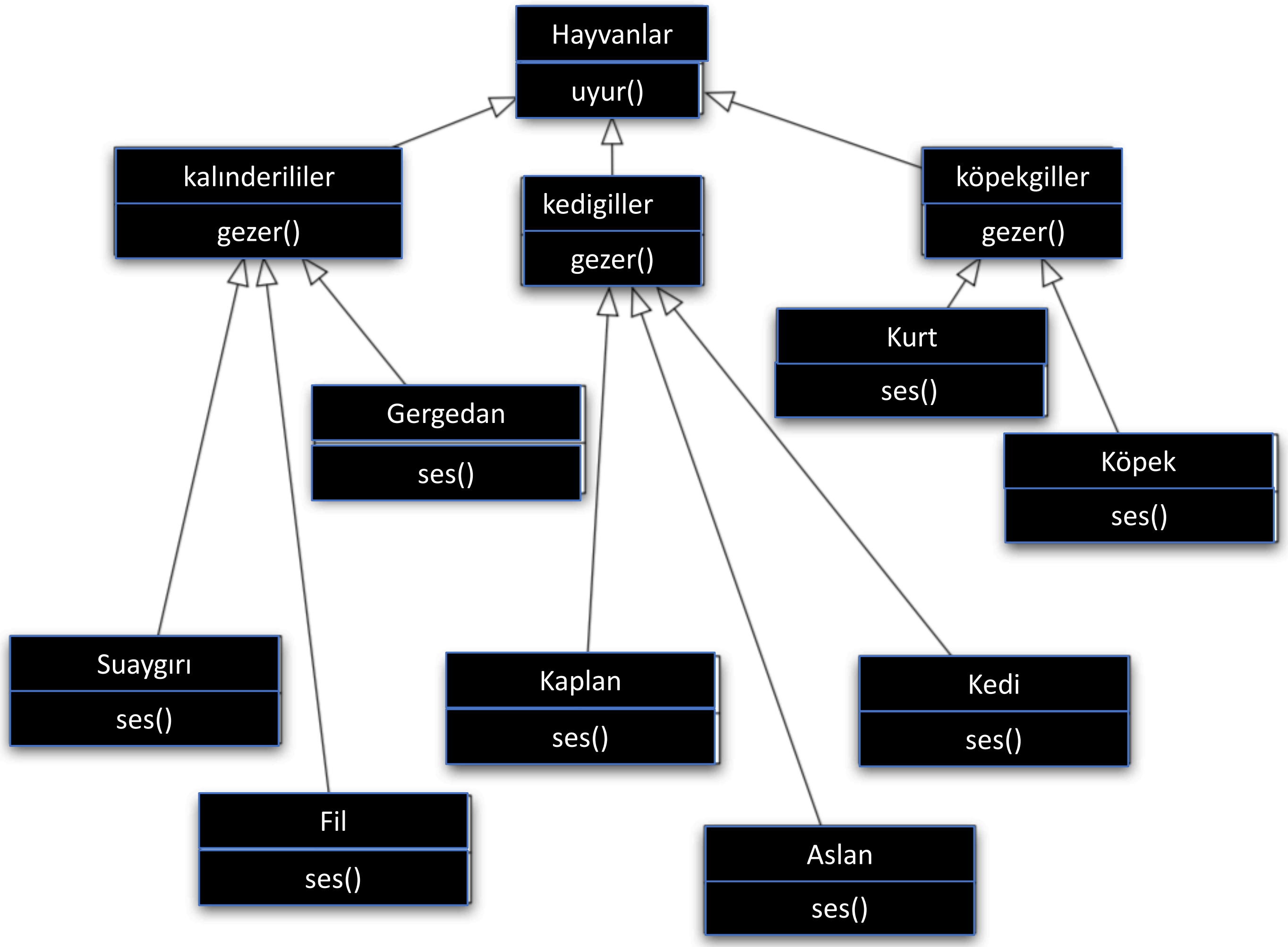
```java
/* File name : RunEncap.java */
public class RunEncap {
      public static void main(String args[]) {
            EncapTest encap = new EncapTest();
      encap.setName("James");
            encap.setAge(20);
            encap.setIdNum("12343ms");
            System.out.print("Name : "
            + encap.getName() + " Age : "
            + encap.getAge());
      }
}
```

https://www.tutorialspoint.com/java/java_encapsulation.htm

Object-Oriented Programming

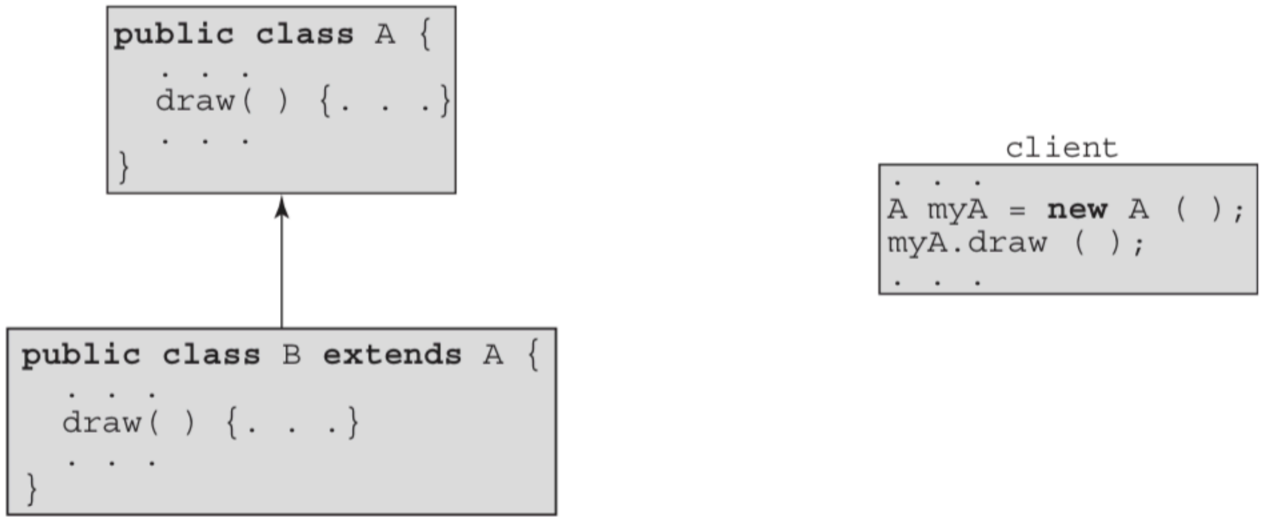
<u>Inheritance (Kalıtım)</u>

- Alt sınıfda bulunan bir varlığın üst sınıfın özelliklerini devralmasına kalıtım denir.



Object-Oriented Programming

- *Dynamic Binding (Polymorphism):* Cok biçimlilik



```
public class A {
   . . .
   draw( ) {. . .}
   . . .
}
```

```
public class B extends A {
   . . .
   draw( ) {. . .}
   . . .
}
```

client

```
. . .
A myA = new A ( );
myA.draw ( );
. . .
```

Object-Oriented Programming

- *Dynamic Binding (Polymorphism):* Cok biçimlilik

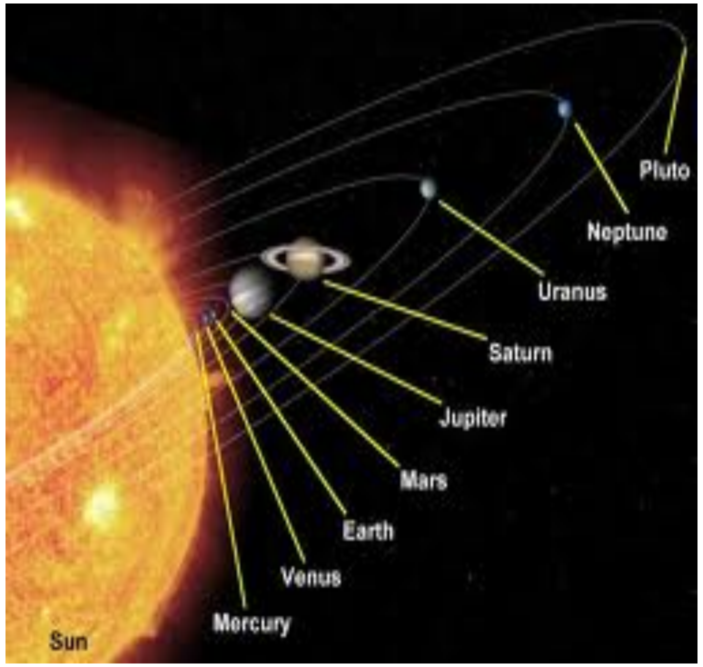
```java
public class Animal{
   public void sound(){
      System.out.println("Animal is making a sound");
   }
}
```

```java
class Horse extends Animal{
    @Override
    public void sound(){
        System.out.println("Neigh");
    }
    public static void main(String args[]){
        Animal obj = new Horse();
        obj.sound();
    }
}
```

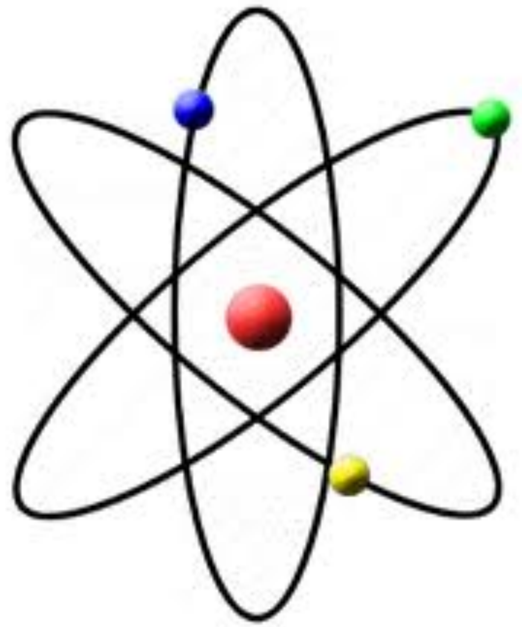
Concurrent Programming

Paralel programlama neden onemli ?

Milky Way Galaxy



Solar System



An Atom

Concurrent Programming

Check in lines

*Traffic Lanes **[20]***

Car Assembly Lanes

C++ : OpenMP API

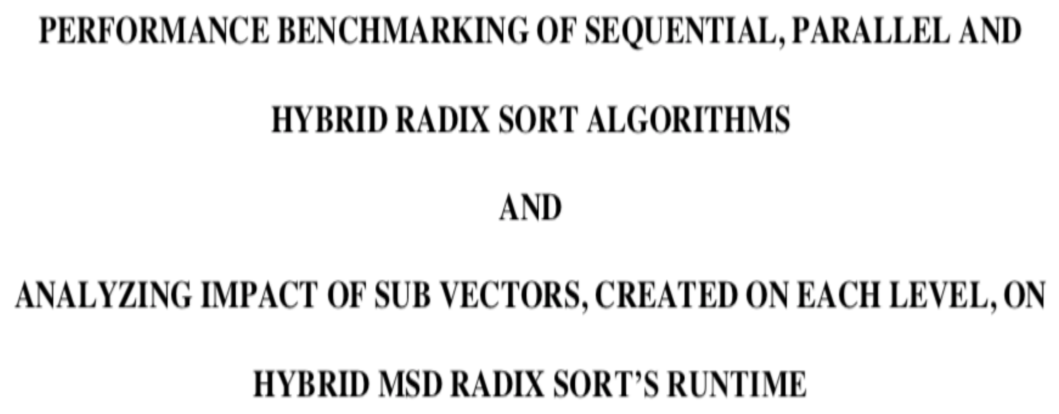
```
#include <omp.h>
```

OpenMP API

- C++ da bulunan paralel programlama için kullanılan kütüphanedir.
  - Compiler Directives,
  - Runtime Library Routines,
  - Environment Variables .

- Single Program Multiple Data (SPMD) Bir program çok veri modelini kullanmakta

C++ : OpenMP API



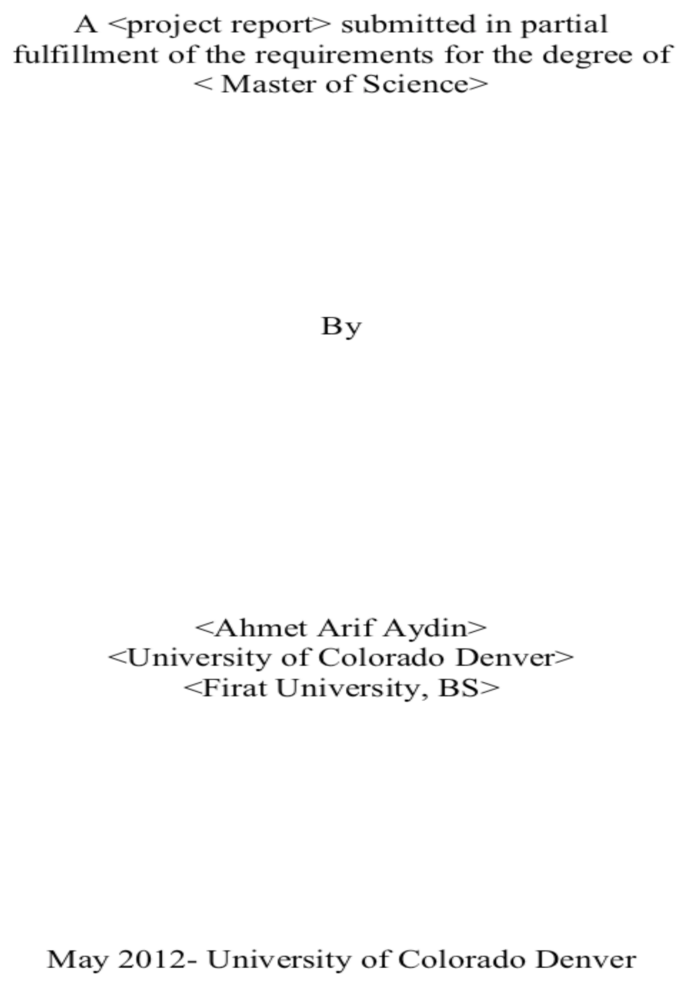
PERFORMANCE BENCHMARKING OF SEQUENTIAL, PARALLEL AND

HYBRID RADIX SORT ALGORITHMS

AND

ANALYZING IMPACT OF SUB VECTORS, CREATED ON EACH LEVEL, ON

HYBRID MSD RADIX SORT'S RUNTIME



A <project report> submitted in partial
fulfillment of the requirements for the degree of
< Master of Science>

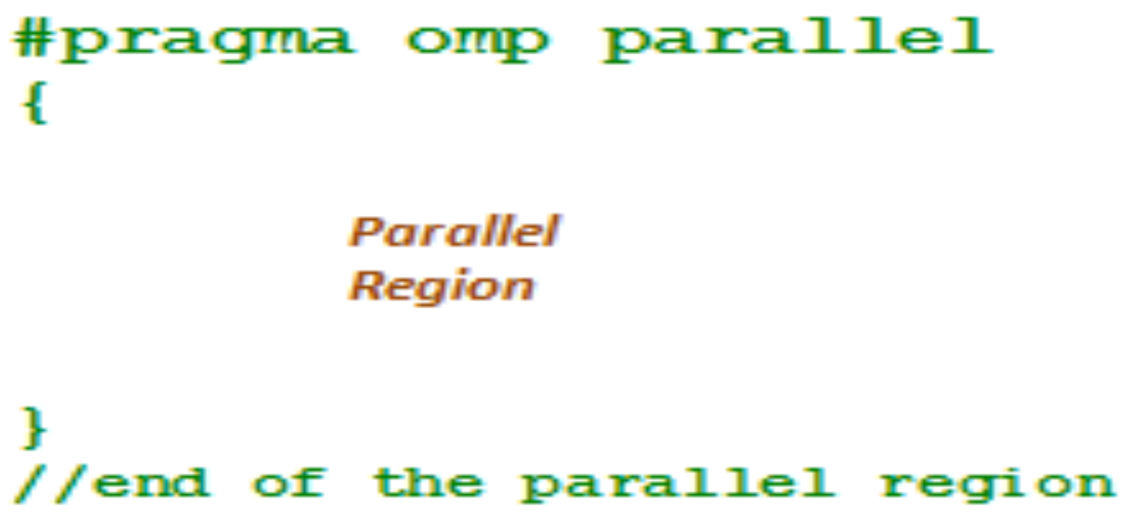
By

<Ahmet Arif Aydin>
<University of Colorado Denver>
<Firat University, BS>

May 2012- University of Colorado Denver

C++ : OpenMP API

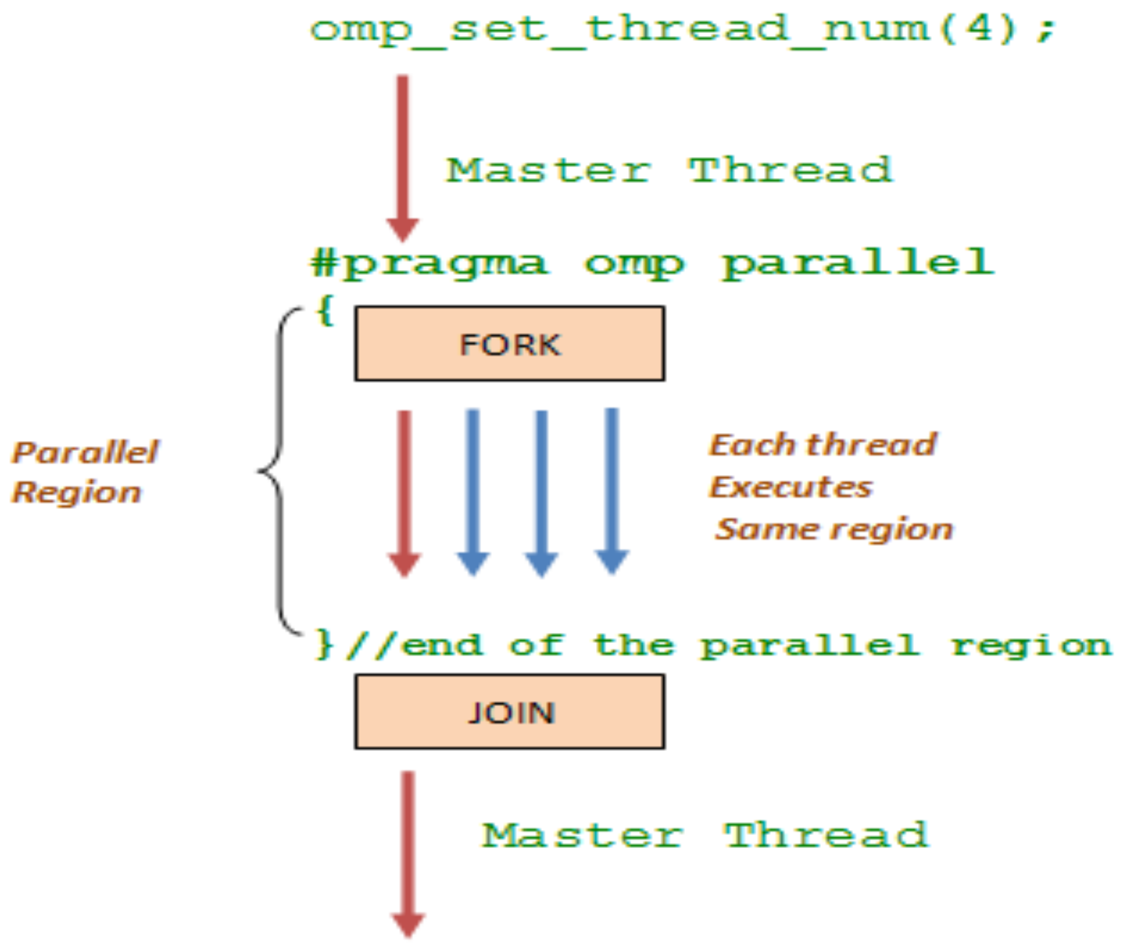
OpenMP API



```
#pragma omp parallel
{

          Parallel
          Region

}
//end of the parallel region
```

C++ : OpenMP API

Fork joın calışma modeli



C++ : OpenMP API

```
#pragma omp parallel
{
        #pragma omp for clauses
        for(int k=0;k<size;k++)
        {
            Each thread executes
            the same program
            simultaneously with
            different data

        }//end of worksharing

}//end of the parallel region
```

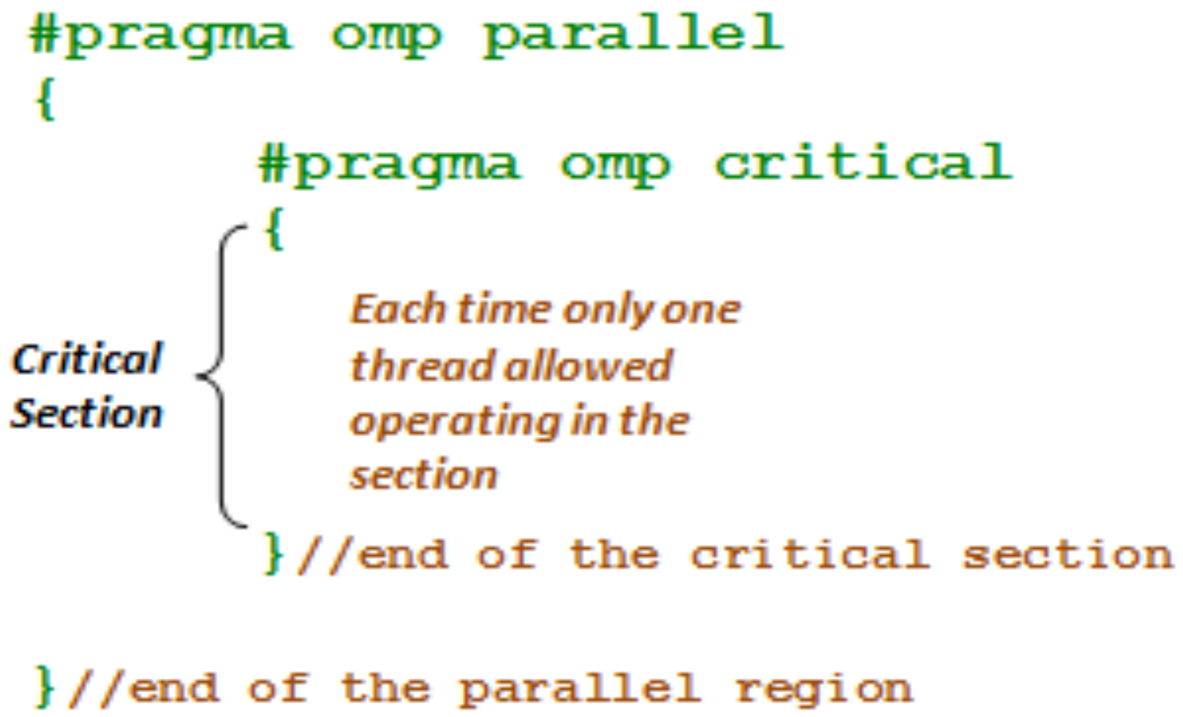
Worksharing Construct

```
schedule(type)
schedule(type,chunksize)
shared(variables)
private(variables)
firstprivate(variables)
lastprivate(variables)
ordered
nowait
```

C++ : OpenMP API

Kritik Bölge

Semafore



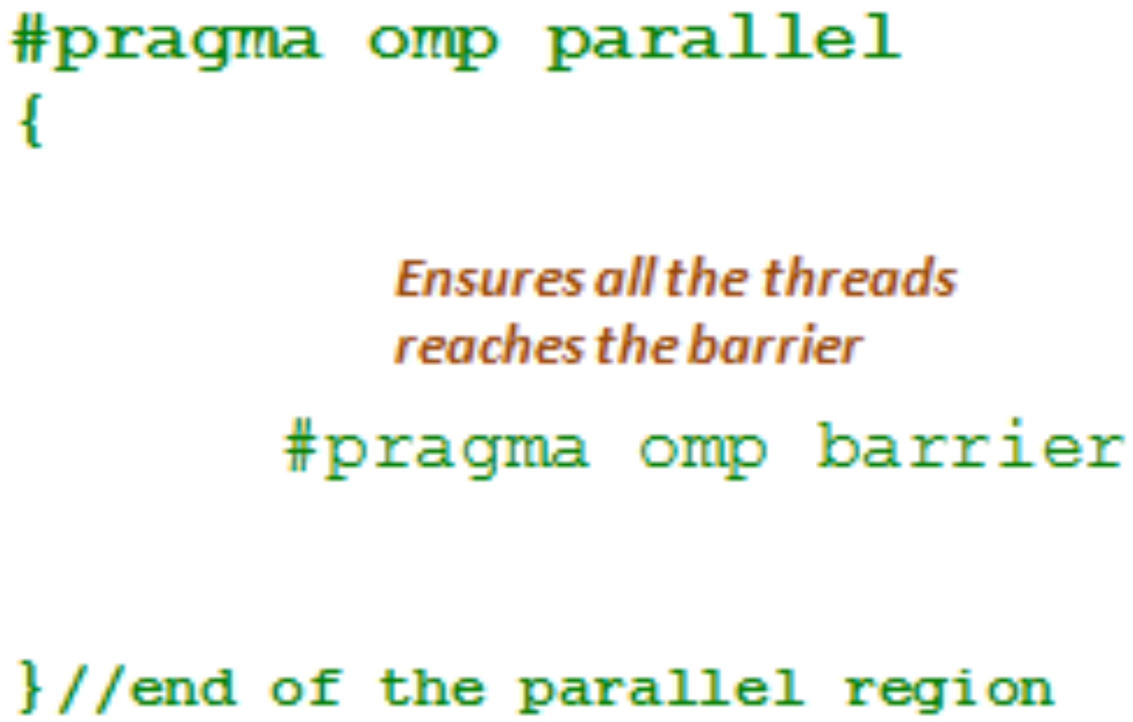
```
#pragma omp parallel
{
        #pragma omp critical
        {

            Each time only one
            thread allowed
            operating in the
            section

        }//end of the critical section

}//end of the parallel region
```

C++ : OpenMP API

```
omp_lock_t lock; // defined a lock variable

omp_init_lock(&lock);//initialized the lock variable

omp_set_thread_num(4);

#pragma omp parallel
{
       #pragma omp for schedule(static) shared(vec) private(i)
       for(int i=0;i<12;i++)
       {
        omp_set_lock(&lock);   //set the lock
        vec[i]=vec[i]* vec[i];
        omp_unset_lock(&lock); //release the lock

       }//end of worksharing

}//end of the parallel region
```

Each time only the thread has the lock allowed operating in the section

C++ : OpenMP API



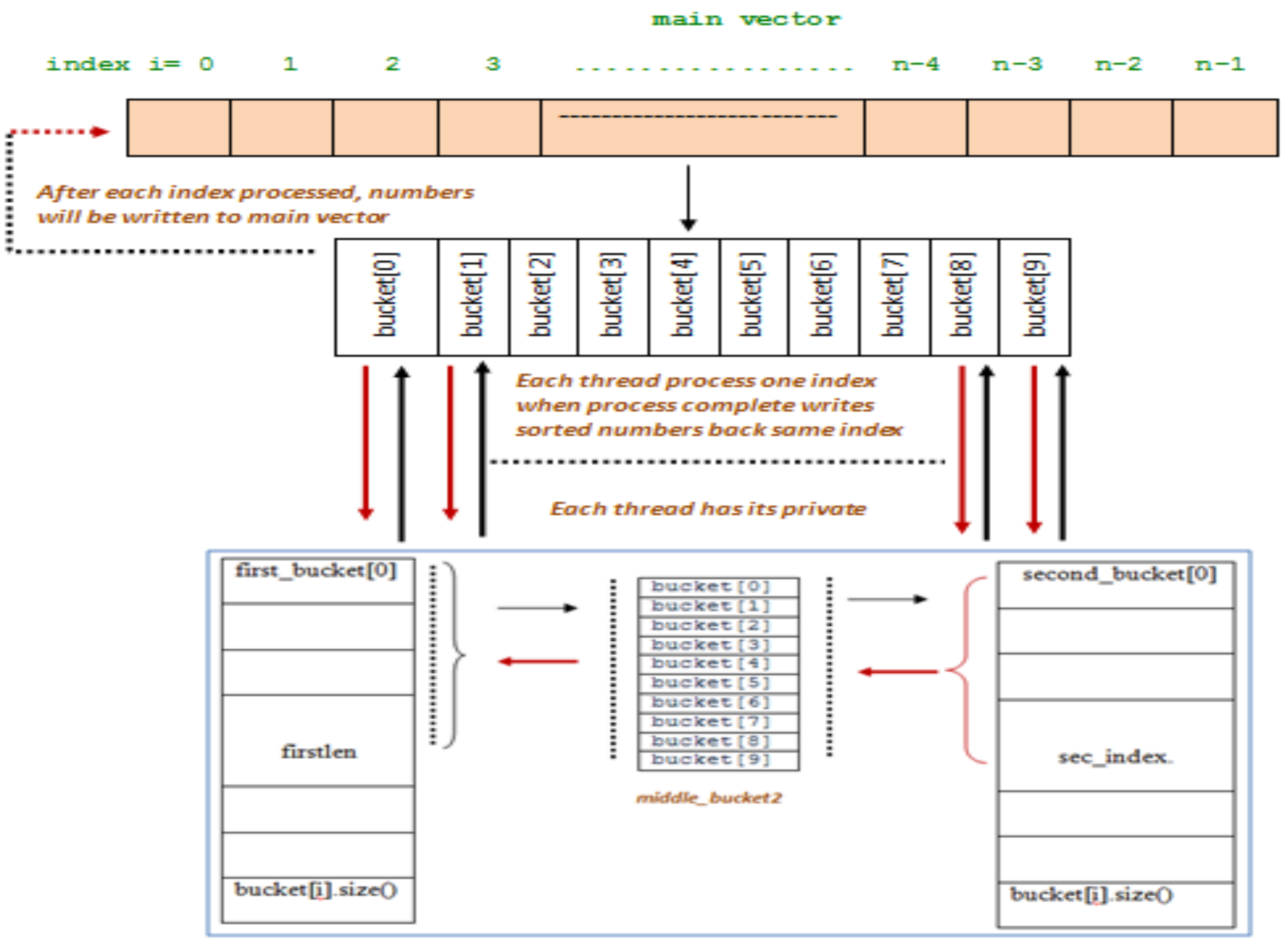
```
#pragma omp parallel
{
```

Ensures all the threads reaches the barrier

```
          #pragma omp barrier


}//end of the parallel region
```

C++ : OpenMP API

MSD Radix Sort


main vector
index i= 0
1
2
3
....
n-4
n-3
n-2
n-1
After each index processed, numbers will be written to main vector
bucket[0]
bucket[1]
bucket[2]
bucket[3]
bucket[4]
bucket[5]
bucket[6]
bucket[7]
bucket[8]
bucket[9]
Each thread process one index when process complete writes sorted numbers back same index
Each thread has its private
first_bucket[0]
firstlen
bucket[i].size()
bucket [0]
bucket [1]
bucket [2]
bucket [3]
bucket [4]
bucket [5]
bucket [6]
bucket [7]
bucket [8]
bucket [9]
middle_bucket2
second_bucket[0]
sec_index
bucket[i].size()