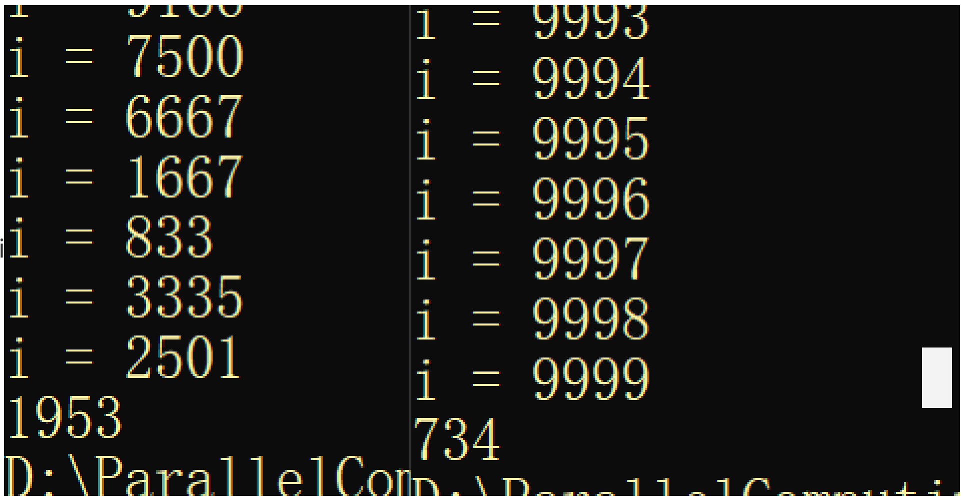
The first try

```
#include<time.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
    int time1=clock(),time2;
#pragma omp parallel for
    for (int i = 0; i < 10000; i++)
    {
        printf("i = %d\n", i);
    }
    time2=clock();
    printf("%d",time2-time1);
    return 0;
}
```

I give it 12 threads, and compare to 1 thread.

I can't believe it, the left is 12 and the right is 1



```
i = 7500
i = 6667
i = 1667
i = 833
i = 3335
i = 2501
1953
D:\ParallelCom
```

```
i = 9994
i = 9995
i = 9996
i = 9997
i = 9998
i = 9999
734
D:\ParallelComputi
```

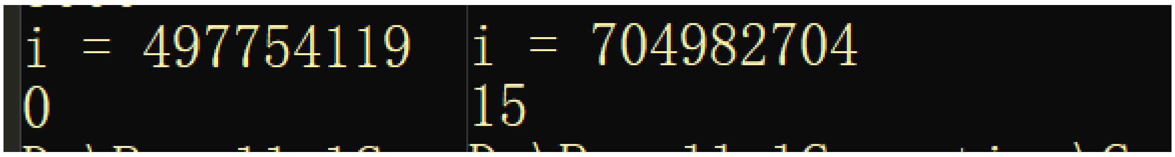
So, parallel computing is not quickly done forever.

The second try

Because the code with print to display, so this time, I will let it output only result

```
#include<time.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
    int time1=clock(),time2;
    int tmp=0;
#pragma omp parallel for
    for (int i = 0; i < 100000; i++)
    {
        tmp+=i;
        if(i%1000==0)
        printf("%d\n",i);
    }
    printf("i = %d\n", tmp);
    time2=clock();
    printf("%d",time2-time1);
    return 0;
}
```

As we guess, the result really good



```
i = 497754119    i = 704982704
0                15
```

because my CPU is very quick, so I have to increase 'i'

We can get the conclusion parallel output use more time than we thought.