Παράδειγμα 1

Δημιουργία ενός αριθμού νημάτων (ο αριθμός καθορίζεται από όρισμα της γραμμής εντολών) και εμφάνιση μηνύματος που περιέχει τον αριθμό νήματος για κάθε νήμα.

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char *argv[]) {
  if (argc != 2) {
    printf("Usage: %s <number_of_threads>\n", argv[0]);
    exit(-1);
  }
  int thread_count = strtol(argv[1], NULL, 10);
#pragma omp parallel num_threads(thread_count)
  {
    int my_rank = omp_get_thread_num();
    int thread_count = omp_get_num_threads();
    printf("Hello from thread %d of %d\n", my_rank, thread_count);
  }
  return 0;
}
```

openmp_example01a.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example01a.c -o openmp_example01a -fopenmp
```

```
./openmp_example01a 4
```

```
Hello from thread 1 of 4
Hello from thread 2 of 4
Hello from thread 0 of 4
Hello from thread 3 of 4
```

ή εναλλακτικά

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

void hello(void);

int main(int argc, char *argv[]) {
  if (argc != 2) {
    printf("Usage: %s <number_of_threads>\n", argv[0]);
    exit(-1);
  }
  int thread_count = strtol(argv[1], NULL, 10);
#pragma omp parallel num_threads(thread_count)
```

```c
  hello();
  return 0;
}

void hello(void) {
  int my_rank = omp_get_thread_num();
  int thread_count = omp_get_num_threads();
  printf("Hello from thread %d of %d\n", my_rank, thread_count);
}
```

openmp_example01b.c

Παράδειγμα 2

Παραδείγματα με μεταβλητές private , firstprivate και shared.

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char *argv[]) {
  int a = 0, b = 0;
#pragma omp parallel num_threads(4) shared(a) firstprivate(b)
  {
    b++;
    int my_rank = omp_get_thread_num();
#pragma omp critical
    a++;

    printf("%d a=%d b=%d\n", my_rank, a, b);
  }
  printf("a=%d b=%d\n", a, b);
  return 0;
}
```

pthreads_example02a.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example02a.c -o openmp_example02a -fopenmp
```

```
./openmp_example02a
```

```
0 a=1 b=1
3 a=2 b=1
1 a=4 b=1
2 a=3 b=1
a=4 b=0
```

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

int main() {
  int a = 1, b = 1, c = 1, d = 1;
  printf("1. a=%d, b=%d, c=%d, d=%d\n", a, b, c, d);

#pragma omp parallel shared(b), private(c), firstprivate(d) num_threads(4)
```

```
  {
    c++;
    d++;
#pragma omp critical
    {
      a++;
      b++;
    }
    printf("2. [%d] a=%d, b=%d, c=%d, d=%d\n", omp_get_thread_num(), a, b, c,
           d);
  }
  printf("3. a=%d, b=%d, c=%d, d=%d\n", a, b, c, d);

  return 0;
}
```

pthreads_example02b.c

```
gcc openmp_example02b.c -o openmp_example02b -fopenmp
```

```
./openmp_example02b
```

```
1. a=1, b=1, c=1, d=1
2. [1] a=2, b=2, c=2292609, d=2
2. [0] a=3, b=3, c=1, d=2
2. [3] a=5, b=5, c=5288825, d=2
2. [2] a=4, b=4, c=5269073, d=2
3. a=5, b=5, c=1, d=1
```

Παράδειγμα 3

Υπολογισμός ορισμένου ολοκληρώματος συνάρτησης (σειριακός και παράλληλος κώδικας).

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

// example f(x)=x^2
double f(double x) { return x * x; }

double trapezio_serial(double a, double b, int n) {
  double x_i, approx;
  int i;
  double h = (b - a) / n;
  approx = (f(a) + f(b)) / 2.0;
  for (i = 1; i <= n - 1; i++) {
    x_i = a + i * h;
    approx += f(x_i);
  }
  approx *= h;
  return approx;
}

void trapezio_parallel(double a, double b, int n, double *global_result) {
  double h, x, my_result, local_a, local_b;
  int i, local_n;
```

```
  int my_rank = omp_get_thread_num();
  int thread_count = omp_get_num_threads();
  h = (b - a) / n;
  local_n = n / thread_count;
  local_a = a + my_rank * local_n * h;
  local_b = local_a + local_n * h;
  my_result = (f(local_a) + f(local_b)) / 2.0;
  for (i = 1; i <= local_n - 1; i++) {
    x = local_a + i * h;
    my_result += f(x);
  }
  my_result *= h;

#pragma omp_critical
  *global_result += my_result;
}

int main(int argc, char **argv) {
  int thread_count = 5;
  double global_result = 0.0;

  int n = 1000000000;
  double approx;
  double a = 0.0, b = 10.0;
  approx = trapezio_serial(a, b, n);
  printf("result from the serial code: %.5f\n", approx);

#pragma omp parallel num_threads(thread_count)
  trapezio_parallel(a, b, n, &global_result);

  printf("result from the parallel code: %.5f\n", global_result);
  return 0;
}
```

openmp_example03a.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example03a.c -o openmp_example03a –fopenmp
```

```
./openmp_example03a
```

```
result from the serial code: 333.33333
result from the parallel code: 333.33333
```

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

// example f(x)=x^2
double f(double x) { return x * x; }

double trapezio_parallel(double a, double b, int n) {
  double h, x, my_result, local_a, local_b;
```

```
  int i, local_n;
  int my_rank = omp_get_thread_num();
  int thread_count = omp_get_num_threads();
  h = (b - a) / n;
  local_n = n / thread_count;
  local_a = a + my_rank * local_n * h;
  local_b = local_a + local_n * h;
  my_result = (f(local_a) + f(local_b)) / 2.0;
  for (i = 1; i <= local_n - 1; i++) {
    x = local_a + i * h;
    my_result += f(x);
  }
  my_result *= h;
  return my_result;
}

int main(int argc, char **argv) {
  double global_result = 0.0;
  int n = 1000000000;
  double a = 0.0, b = 10.0;

#pragma omp parallel num_threads(5)
  {
    double my_result = trapezio_parallel(a, b, n);
#pragma omp_critical
    global_result += my_result;
  }
  printf("result from the parallel code: %.5f\n", global_result);
  return 0;
}
```

openmp_example03b.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example03b.c -o openmp_example03b –fopenmp
./openmp_example03b
result from the parallel code: 333.33333
```

Υπολογισμός ορισμένου ολοκληρώματος συνάρτησης με χρήση reduction.

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

// example f(x)=x^2
double f(double x) { return x * x; }

double trapezio_parallel(double a, double b, int n) {
  double h, x, my_result, local_a, local_b;
  int i, local_n;
  int my_rank = omp_get_thread_num();
  int thread_count = omp_get_num_threads();
  h = (b - a) / n;
```

```
  local_n = n / thread_count;
  local_a = a + my_rank * local_n * h;
  local_b = local_a + local_n * h;
  my_result = (f(local_a) + f(local_b)) / 2.0;
  for (i = 1; i <= local_n - 1; i++) {
    x = local_a + i * h;
    my_result += f(x);
  }
  my_result *= h;
  return my_result;
}

int main(int argc, char **argv) {
  double global_result = 0.0;
  int n = 1000000000;
  double a = 0.0, b = 10.0;

#pragma omp parallel num_threads(5) reduction(+ : global_result)
  global_result = trapezio_parallel(a, b, n);

  printf("result from the parallel code using reduction : %.5f\n",
         global_result);
  return 0;
}
```

openmp_example03c.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example03c.c -o openmp_example03c –fopenmp
```

```
./openmp_example03c
```

```
result from the parallel code using reduction : 333.33333
```

Παράδειγμα 4

Δομές διαμοιρασμού εργασίας (single, sections, for)

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

// Δομές διαμοιρασμού εργασίας (single, sections, for)

int main() {
  int i;

#pragma omp parallel num_threads(5)
  {
    // ο ακόλουθος κώδικας θα εκτελεστεί από όλα τα νήματα
    printf("Hi from thread %d\n", omp_get_thread_num());

// ο ακόλουθος κώδικας θα εκτελεστεί από ένα μόνο τυχαίο νήμα
#pragma omp single
    printf("The thread chosen for single execution is %d\n",
           omp_get_thread_num());
```

```c
// δύο τυχαία νήματα θα εκτελέσουν από ένα από τα ακόλουθα sections
#pragma omp sections
    {
#pragma omp section
      { printf("Section A was executed by %d\n", omp_get_thread_num()); }
#pragma omp section
      { printf("Section B was executed by %d\n", omp_get_thread_num()); }
    }

// οι 10 επαναλήψεις της for θα μοιραστούν από 2 σε κάθε ένα από τα 5 νήματα
#pragma omp for
    for (i = 0; i < 10; i++)
      printf("Iteration %d is given to thread %d\n", i, omp_get_thread_num());
  }
}
```

openmp_example04a.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example04a.c -o openmp_example04a –fopenmp
```

```
./openmp_example04a
```

```
Hi from thread 3
The thread chosen for single execution is 3
Hi from thread 0
Hi from thread 2
Hi from thread 4
Hi from thread 1
Section A was executed by 4
Section B was executed by 3
Iteration 4 is given to thread 2
Iteration 5 is given to thread 2
Iteration 2 is given to thread 1
Iteration 3 is given to thread 1
Iteration 6 is given to thread 3
Iteration 7 is given to thread 3
Iteration 8 is given to thread 4
Iteration 9 is given to thread 4
Iteration 0 is given to thread 0
Iteration 1 is given to thread 0
```

Παράδειγμα 5

Υπολογισμός του αριθμού π με βάση τον τύπο:

$$\int_0^1 \frac{4}{1+x^2} dx = \pi$$

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

#define n 1000000000
```

```
double compute_pi_serial() {
  int i = 0;
  double pi = 0.0, w = 1.0 / n;
  for (i = 0; i < n; i++)
    pi += 4 * w / (1 + (i + 0.5) * (i + 0.5) * w * w);
  return pi;
}

void compute_pi_parallel() {
  int i;
  double pi = 0.0, mysum = 0.0, w = 1.0 / n;
#pragma omp parallel firstprivate(mysum) num_threads(1000)
  {
#pragma omp for
    for (i = 0; i < n; i++)
      mysum += 4 * w / (1 + (i + 0.5) * (i + 0.5) * w * w);

#pragma omp critical
    pi += mysum;
  }
  printf("Pi computed in parallel using 10^9 terms: %.14f\n", pi);
}

void compute_pi_parallel_using_reduction() {
  int i;
  double pi = 0.0, w = 1.0 / n;
#pragma omp parallel for reduction(+ : pi)
  for (i = 0; i < n; i++)
    pi += 4 * w / (1 + (i + 0.5) * (i + 0.5) * w * w);

  printf("Pi computed in parallel(reduction) using 10^9 terms: %.14f\n", pi);
}

int main() {
  double start, finish;

  start = omp_get_wtime();
  printf("Pi computed serially using 10^9 terms  : %.14f\n",
         compute_pi_serial());
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f\n", finish - start);

  start = omp_get_wtime();
  compute_pi_parallel();
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f\n", finish - start);

  start = omp_get_wtime();
  compute_pi_parallel_using_reduction();
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f\n", finish - start);
```

```
  return 0;
}
```

openmp_example05a.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example05a.c -o openmp_example05a –fopenmp
```

```
./openmp_example05a
```

```
Pi computed serially using 10^9 terms  : 3.14159265358938
Time elapsed: 7.20
Pi computed in parallel using 10^9 terms: 3.14159265358979
Time elapsed: 1.95
Pi computed in parallel(reduction) using 10^9 terms: 3.14159265358978
Time elapsed: 1.95
```

Υπολογισμός του π με βάση τον τύπο:

$$\pi = 4\left[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right] = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}$$

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

#define n 1000000000

double compute_pi_serial() {
  int k;
  double factor = 1.0;
  double sum = 0.0;
  for (k = 0; k < n; k++) {
    sum += factor / (2 * k + 1);
    factor = -factor;
  }
  return 4.0 * sum;
}

void compute_pi_parallel() {
  int k;
  double sum = 0.0, factor;
#pragma omp parallel for reduction(+ : sum) private(factor)
  for (k = 0; k < n; k++) {
    if (k % 2 == 0)
      factor = 1.0;
    else
      factor = -1.0;
    sum += factor / (2 * k + 1);
  }
  printf("Pi computed in parallel using 10^9 terms: %.14f\n", 4.0 * sum);
}

int main() {
  double start, finish;
```

```
  start = omp_get_wtime();
  printf("Pi computed serially using 10^9 terms  : %.14f\n",
         compute_pi_serial());
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f\n", finish - start);

  start = omp_get_wtime();
  compute_pi_parallel();
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f\n", finish - start);

  return 0;
}
```

openmp_example05b.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example05b.c -o openmp_example05b –fopenmp
```

```
./openmp_example05b
```

```
Pi computed serially using 10^9 terms  : 3.14159265258805
Time elapsed: 7.30
Pi computed in parallel using 10^9 terms: 3.14159265258932
Time elapsed: 1.93
```

Παράδειγμα 6

Χρονοπρογραμματισμός βρόχων. Επιλογή χρονοδιαγραμμάτων.

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

// n=30000
// serial execution: 11.02 seconds
// parallel for : 5.83 seconds
// parallel for schedule(static, 1): 4.61 seconds

double f(int i) {
  int j, start = i * (i + 1) / 2, finish = start + i;
  double return_val = 0;
  for (j = start; j < finish; j++)
    return_val += sin(j);
  return return_val;
}

int main() {
  double sum;
  int i, n = 30000;
  double start, finish;

  start = omp_get_wtime();
  sum = 0;
```

```
  for (i = 0; i <= n; i++)
    sum += f(i);
  printf("The sum is %.2f\n", sum);
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f (serial)\n", finish - start);

  start = omp_get_wtime();
  sum = 0;
// χρονοδιάγραμμα τύπου μπλοκ
#pragma omp parallel for num_threads(4) reduction(+ : sum)
  for (i = 0; i <= n; i++)
    sum += f(i);
  printf("The sum is %.2f\n", sum);
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f (default schedule)\n", finish - start);

  start = omp_get_wtime();
  sum = 0;
// κυκλικό χρονοδιάγραμμα
#pragma omp parallel for num_threads(4) reduction(+ : sum) schedule(static, 1)
  for (i = 0; i <= n; i++)
    sum += f(i);
  printf("The sum is %.2f\n", sum);
  finish = omp_get_wtime();
  printf("Time elapsed: %.2f (cyclic schedule)\n", finish - start);
  return 0;
}
```

openmp_example06.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example06.c -o openmp_example06 –fopenmp
```

```
./openmp_example06
```

```
The sum is 6.49
Time elapsed: 20.83 (serial)
The sum is 6.49
Time elapsed: 9.58 (default schedule)
The sum is 6.49
Time elapsed: 5.85 (cyclic schedule)
```

Παράδειγμα 7

Ταξινόμηση

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

void bubble_sort_serial(int *a, int n) {
  int i, list_length, tmp;
  for (list_length = n; list_length >= 2; list_length--)
    for (i = 0; i < list_length - 1; i++)
      if (a[i] > a[i + 1]) {
```

```
        tmp = a[i];
        a[i] = a[i + 1];
        a[i + 1] = tmp;
      }
}

void odd_even_sort_serial(int *a, int n) {
  int i, phase, tmp;
  for (phase = 0; phase < n; phase++)
    if (phase % 2 == 0) {
      for (i = 1; i < n; i += 2)
        if (a[i - 1] > a[i]) {
          tmp = a[i];
          a[i] = a[i - 1];
          a[i - 1] = tmp;
        }
    } else {
      for (i = 1; i < n - 1; i += 2)
        if (a[i] > a[i + 1]) {
          tmp = a[i];
          a[i] = a[i + 1];
          a[i + 1] = tmp;
        }
    }
}

void odd_even_sort_parallel1(int *a, int n, int thread_count) {
  int i, phase, tmp;
  for (phase = 0; phase < n; phase++)
    if (phase % 2 == 0) {
#pragma omp parallel for num_threads(thread_count) default(none)               \
    shared(a, n) private(i, tmp)
      for (i = 1; i < n; i += 2)
        if (a[i - 1] > a[i]) {
          tmp = a[i];
          a[i] = a[i - 1];
          a[i - 1] = tmp;
        }
    } else {
#pragma omp parallel for num_threads(thread_count) default(none)               \
    shared(a, n) private(i, tmp)
      for (i = 1; i < n - 1; i += 2)
        if (a[i] > a[i + 1]) {
          tmp = a[i];
          a[i] = a[i + 1];
          a[i + 1] = tmp;
        }
    }
}

void odd_even_sort_parallel2(int *a, int n, int thread_count) {
  int i, phase, tmp;
```

```
#pragma omp parallel num_threads(thread_count) default(none)                    \
    shared(a, n) private(i, tmp, phase)
  for (phase = 0; phase < n; phase++)
    if (phase % 2 == 0) {
#pragma omp for
      for (i = 1; i < n; i += 2)
        if (a[i - 1] > a[i]) {
          tmp = a[i];
          a[i] = a[i - 1];
          a[i - 1] = tmp;
        }
    } else {
#pragma omp for
      for (i = 1; i < n - 1; i += 2)
        if (a[i] > a[i + 1]) {
          tmp = a[i];
          a[i] = a[i + 1];
          a[i + 1] = tmp;
        }
    }
}

int main() {
  int i, n = 40000;
  int a[n], c[n];
  double start, finish;
  srand(12345L);

  for (i = 0; i < n; i++)
    c[i] = a[i] = rand() % 1000000;
  start = omp_get_wtime();
  bubble_sort_serial(a, n);
  finish = omp_get_wtime();
  printf("Time elapsed bubble sort serial %.2f seconds\n", finish - start);

  for (i = 0; i < n; i++)
    a[i] = c[i];
  start = omp_get_wtime();
  odd_even_sort_serial(a, n);
  finish = omp_get_wtime();
  printf("Time elapsed odd even sort serial %.2f seconds\n", finish - start);

  for (i = 0; i < n; i++)
    a[i] = c[i];
  start = omp_get_wtime();
  odd_even_sort_parallel1(a, n, 4);
  finish = omp_get_wtime();
  printf("Time elapsed odd even sort parallel1 %.2f seconds\n", finish - start);

  for (i = 0; i < n; i++)
    a[i] = c[i];
  start = omp_get_wtime();
```

```
  odd_even_sort_parallel2(a, n, 4);
  finish = omp_get_wtime();
  printf("Time elapsed odd even sort parallel2 %.2f seconds\n", finish - start);

  return 0;
}
```

openmp_example07.c

Μεταγλώττιση – εκτέλεση – έξοδος

```
gcc openmp_example07.c -o openmp_example07 –fopenmp
```

```
./openmp_example07
```

```
Time elapsed bubble sort serial 8.18 seconds
Time elapsed odd even sort serial 6.62 seconds
Time elapsed odd even sort parallel1 3.73 seconds
Time elapsed odd even sort parallel2 2.73 seconds
```