```cpp
//--------------------------------------------------------------------------
// Computation of pi by the method of Monte Carlo
//--------------------------------------------------------------------------
#if (defined __linux__) || (defined _AIX)
   #include <sys/types.h>
   #include <sys/stat.h>
   #include <unistd.h>
#elif (defined _WIN32) || (defined _WIN64)
   #include <conio.h>
   #include <direct.h>
#endif

#include <mpi.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <iostream>
using namespace std;
//--------------------------------------------------------------------------

#define pocet_bodov_celkom 1000000000

int Monte_Carlo(int, int);

int main(int argc, char** argv)
{
   int i, rank, size, pocet_bodov_pre_proces, pocet_bodov_v_kruhu_z_procesu,
      pocet_bodov_v_kruhu_celkom = 0;
   double pi, my_time;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   if (rank == 0) {           // rank 0 is a master
      cout << "\nCelkovy poccet bodov: " << pocet_bodov_celkom << "\n";
      cout << "Pocet procesov: 1 master + " << size - 1 << " slaves\n\n";
   }
   fflush(stdout);
   MPI_Barrier(MPI_COMM_WORLD); // DISPLAY TASK MANAGER!!!
   my_time = -MPI_Wtime();

   if (rank < size - 1)
      pocet_bodov_pre_proces = pocet_bodov_celkom / size;
   else
      pocet_bodov_pre_proces = pocet_bodov_celkom - (size - 1)
         * (pocet_bodov_celkom / size);

   pocet_bodov_v_kruhu_z_procesu = Monte_Carlo(pocet_bodov_pre_proces, rank);
   if (rank == 0)
      pocet_bodov_v_kruhu_celkom = pocet_bodov_v_kruhu_z_procesu;
   MPI_Barrier(MPI_COMM_WORLD);

   if (size > 1)
```

```cpp
      if (rank > 0)
         MPI_Ssend(&pocet_bodov_v_kruhu_z_procesu, 1, MPI_INT, 0, rank,
            MPI_COMM_WORLD);
      else
         for (i = 1; i < size; i++) {
            MPI_Recv(&pocet_bodov_v_kruhu_z_procesu, 1, MPI_INT, i, i,
               MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            pocet_bodov_v_kruhu_celkom += pocet_bodov_v_kruhu_z_procesu;
         }

   MPI_Barrier(MPI_COMM_WORLD);
   cout << "proces " << rank << ": pocet bodov = " << pocet_bodov_pre_proces
      << ", pocet bodov vo vnutri kruhu = " << pocet_bodov_v_kruhu_z_procesu << "\n";

   // kruh s polomerom 1 je vpisany do stvorca
   // Plocha kruhu PK = pi*1*1 = pi
   // Plocha stvorca PS = 2*2 = 4
   //
   // Trojclenka:
   // pi / 4 = pocet_bodov_v_kruhu_celkom / pocet_bodov_celkom (staci brat do uvahy
   // len 1-vy kvadrant)
   //
   // pi = pocet_bodov_v_kruhu_celkom * 4 / pocet_bodov_celkom

   if (rank == 0) {
      pi = (pocet_bodov_v_kruhu_celkom * 4.0) / pocet_bodov_celkom;
      printf("\nVysledne pi = %10.10f\n", pi);
   }

   my_time += MPI_Wtime();
   MPI_Barrier(MPI_COMM_WORLD);
   fflush(stdout);
   if (rank == 0)
      printf("\nTime = %10.6f s\n", my_time);

   MPI_Finalize();

   return 0;
}
//---------------------------------------------------------------------------
int Monte_Carlo(int pocet_v_stvorci, int rank)
{
   int i, pocet_bodov_v_kruhu_z_procesu = 0;
   double x = 0, y = 0;

   srand((unsigned short)time(NULL) + (unsigned short)rank * 100);
   for (i = 0; i < pocet_v_stvorci; i++) {
      x = (double)rand() / RAND_MAX;
      y = (double)rand() / RAND_MAX;
      if (x * x + y * y <= 1)
         pocet_bodov_v_kruhu_z_procesu++;
   }
   return pocet_bodov_v_kruhu_z_procesu;
```

```
107 }
108 //---------------------------------------------------------------------------
109
```