Príloha 2 - Gaussova eliminácia

Implementácia na jednom procesore

```
void GSolve(double **a,int n,double *x)
{
  int i,j,k;
  double tmp;

  for (k = 0; k < n - 1; k++)
    for (i = k + 1; i < n; i++)  {
      a[i][k] = a[i][k] / a[k][k];
      for (j = k + 1; j < n + 1; j++)
        a[i][j] = a[i][j] - a[i][k] * a[k][j];
    }

  //backwards substitution
  for (i = n - 1; i >= 0; i--) {
    tmp = a[i][n];
    for (j = i + 1; j < n; j++)
      tmp -= a[i][j] * x[j];
    x[i] = tmp / a[i][i];
  }
}
```

Implementácia pomocou OpenMP

```
#define PROC(x) ((x) % nid)
#define MAP(x) (PROC(x) == id)
void GSolve(double **a,int n,double *x) {
  int i,j,k;
  double tmp;

 #pragma omp parallel shared (a)
 {
  id = omp_get_thread_num();
  nid = omp_get_num_threads();

  for (k = 0; k < n - 1; k++)  {
    for (i = k + 1; i < n; i++)  {
      //only if we're the processor that handles i
      if (MAP(i)) {
        a[i][k] = a[i][k] / a[k][k];
        for (j = k + 1; j < n + 1; j++)
          a[i][j] = a[i][j] - a[i][k] * a[k][j];
      }
    }
  }
 }

  //backwards substitution
  for (i = n - 1; i >= 0; i--) {
    tmp = a[i][n];
    for (j = i + 1; j < n; j++)
      tmp -= a[i][j] * x[j];
    x[i] = tmp / a[i][i];
  }
}
```

Implementácia pomocou MPI

```
#define PROC(x) ((x) % nid)
#define MAP(x) (PROC(x) == id)
void GSolve(double **a,int n,double *x)
{
  int i,j,k;
  double tmp;

  //Is row 0 assigned to me? Broadcast row 0 if yes, receive it otherwise.
  MPI_Bcast ((char *) a[0], rowsize, MPI_CHAR, PROC(0), MPI_COMM_WORLD);
```

```
  for (k = 0; k < n - 1; k++) {
    for (i = k + 1; i < n; i++) {
      //only if we're the processor that handles i
      if (MAP(i)) {
        a[i][k] = a[i][k] / a[k][k];
        for (j = k + 1; j < n + 1; j++)
          a[i][j] = a[i][j] - a[i][k] * a[k][j];
      }
    }
    //if I am assigned to k+1, broadcast row k+1, else receive it
    MPI_Bcast ((char *) a[k+1], rowsize, MPI_CHAR, PROC(k+1), MPI_COMM_WORLD);
  }

  //backwards substitution
  for (i = n - 1; i >= 0; i--) {
    if (id == 0) {
      if (!MAP(i))
        //receive this row
        MPI_Recv ((char *) a[i], rowsize, MPI_CHAR, PROC(i), 0,
            MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      tmp = a[i][n];
      for (j = i + 1; j < n; j++)
        tmp -= a[i][j] * x[j];
      x[i] = tmp / a[i][i];
    }
    else
      if (MAP(i))
        //send this row to process 0
        MPI_Send ((char *) a[i], rowsize, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
  }
}


Hlavný program - posielanie údajov procesorom:
if (id == 0) {
  for (i = 0; i < SIZE; i++)
  {
    for (j = 0; j < SIZE + 1; j++)
      a[i][j] = (double) (random() % 1000 + 1);
    if (PROC(i) != 0)
      //send the row to its processor
      MPI_Send ((char *) a[i], rowsize, MPI_CHAR, PROC(i), 0,
          MPI_COMM_WORLD);
  }
}
else
  for (i = 0; i < SIZE; i++)
    if (MAP(i))
      //receive this row from processor 0
      MPI_Recv ((char *) a[i], rowsize, MPI_CHAR, 0, 0,
          MPI_COMM_WORLD, MPI_STATUS_IGNORE);
GSolve (a, SIZE, x);
```