Práctica 4.

MPI: Tipos de Datos Derivados

MPI maneja en todas sus funciones de envío/recepción vectores de tipos simples. En general, se asume que los elementos de esos vectores están almacenados consecutivamente en memoria. En ocasiones, sin embargo, es necesario el intercambio de tipos estructurados, o de vectores no almacenados consecutivamente en memoria por ejemplo el envío de una columna de un arreglo, en vez de envío de una fila.

MPI incluye la posibilidad de definir tipos más complejos (objetos del tipo MPI_Datatype), empleando constructores. Antes de usar un tipo de usuario, hay que ejecutar MPI_Type_commit(). Cuando ya no se necesite un tipo, se puede liberar con MPI_Type_free().

```
int MPI_Type_commit(MPI_Datatype *datatype);

int MPI_Type_free(MPI_Datatype *datatype);
```

Definición de tipos homogéneos

Son tipos homogéneos aquellos tipos en los que todos los elementos constituyentes son del mismo tipo. Se pueden definir tipos homogéneos con dos funciones distintas: MPI_Type_contiguous() y MPI_Type_vector(). La primera versión es la más sencilla, y permite definir un tipo formado por una colección de elementos de un tipo básico, todos ellos del mismo tamaño y almacenados consecutivamente en memoria.

```
int MPI_Type_contiguous (int count, MPI_Datatype oldtype,
                         MPI_Datatype *newtype);
```

newtype es un nuevo tipo que consiste en **count** copias de **oldtype**. Si los elementos constituyentes del nuevo tipo no están almacenados consecutivamente en memoria, sino que están espaciados a intervalos regulares, la función a emplear es MPI_Type_vector().

```
int MPI_Type_vector (int count, int blocklength, int
stride, MPI_Datatype oldtype, MPI_Datatype *newtype);
```

newtype es un nuevo tipo que consiste en **count** bloques de datos. Cada bloque consta de **blocklength** elementos del tipo **oldtype**. La distancia entre bloques, medida en múltiplos del tamaño del elemento básico, la da **stride**.
Una llamada a MPI_Type_contiguous(c, o, n) equivale a una llamada a MPI_Type_vector (c, 1, 1, o, n), o a una llamada a MPI_Type_vector (1, c, x, o, n), siendo x un valor arbitrario.

Ejercicios

1. Defina un tipo de datos derivado que permita enviar una columna de la matriz a los distintos procesos. Cómo hacemos para enviar dos columnas consecutivas usando el tipo de datos definido?.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

2. Defina un tipo de datos derivado que permita enviar dos columnas por vez a los distintos procesos

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

3. Defina un tipo de datos derivado que permita enviar la diagonal principal de la matriz a un proceso

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

4. Defina un tipo de datos derivado que permita enviar la anti-diagonal a un proceso

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

5. Defina un tipo de datos derivado que permita enviar los elementos de la matriz usando el siguiente patrón.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

6. Defina un tipo de datos derivado que permita enviar los elementos de la matriz usando el siguiente patrón.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

```
/* ---------------------------------------------------------------------
 * Code:   choose.c
 * Lab:    MPI Derived Datatypes
 *
 * Usage:  choose
 * --------------------------------------------------------------------- */

#include "mpi.h"

#define IMAX 16                    /* numero de filas                    */
#define JMAX 16                    /* numero de columnas                 */
#define SOURCERANK 0               /* rango del proceso Fuente           */

main( int argc, char **argv ) {

  int i, j, myrank;
  int count=0;

  MPI_Status status ;
  MPI_Datatype coltype ;

  int matrix [IMAX][JMAX] ;
  /* Inicializacion del ambiente MPI   */

  MPI_Init( &argc, &argv ) ;
  MPI_Comm_rank( MPI_COMM_WORLD, &myrank ) ;
  printf ( "Proceso %d inicializado \n", myrank ) ;

  /* Inicializacion de la matriz       */
  if ( myrank == SOURCERANK )
    for ( i=0; i<IMAX; i++)
      for (j=0; j<JMAX; j++)
         matrix[i][j] = count++;
  else
    for ( i=0; i<IMAX; i++)
      for (j=0; j<JMAX; j++)
         matrix[i][j] = 0;


  /* Definicion del nuevo tipo de datos derivado segun el ejercicio  */



  if ( myrank == SOURCERANK ) {

    /* El proceso SOURCERANK reparte la matriz usando el nuevo tipo de dato
derivado definido */                                          */


  }

  else  {
   /* el resto de los procesos recibe la porción de la matriz que le
corresponde */
   /* e imprime la matriz completa                                  */


  }
```