Tutorial/Lab Session 4

PURPOSE:

1. To practice the use of data structure with C programming language.
2. To practice the use of matrix with C programming language.

PROCEDURE:

Practice 1: To write a data structure into a file.

Step 1: Login to the PC and start the X-window environment.

Step 2: Create a directory called "lab4" (use "mkdir lab4").

Step 3: Go to the directory lab4 (use "cd lab4").

Step 4: Edit the following two files: savedata.c and Makefile.

```
#include <stdio.h>
#include <string.h>
typedef struct {
        char    name[100] ;
        double  height, weight ;
}       PersonalData ;
main(int argc, char **argv)
{
        FILE            *pf ;
        PersonalData    jimmy ;
        strcpy(jimmy.name, "Jimmy Lim") ;
        jimmy.height = 1.8 ;
        jimmy.weight = 60.0 ;
        pf = fopen("jimmy.dat", "w") ;
        if (pf == NULL) exit(0) ;
        fwrite((char *) &jimmy, sizeof(PersonnalData),
                1, pf) ;
        fclose(pf) ;
}
```

savedata.c

```
.c.o:
        cc  -c   $*.c

savedata:   savedata.o
        cc -o    savedata   savedata.o
```

Makefile

Step 5: Compile the program (use "make savedata").

Step 6: Execute the program (type "savedata").

Practice 2: To read a data structure from a file.

Step 1: Edit the following two files: readdata.c and Makefile.

```
#include <stdio.h>
#include <string.h>
typedef struct {
        char    name[100] ;
        double  height, weight ;
}       PersonalData ;
main(int argc, char **argv)
{
        FILE            *pf ;
        PersonalData    jimmy ;
        pf = fopen("jimmy.dat", "r") ;
        if (pf == NULL) exit(0) ;
        fread((char *) &jimmy, sizeof(PersonnalData),
                1, pf) ;
        fclose(pf) ;
        printf("\n Jimmy's name: %s", jimmy.name) ;
        printf("\n Jimmy's height: %lf", jimmy.height);
        print("\n Jimmy's weight: %lf", jimmy.weight);
}
```

readdata.c

```
.c.o:
        cc  -c   $*.c

readdata:   readdata.o
        cc -o    readdata   readdata.o
```

Makefile

Step 2: Compile the program (use "make readdata").

Step 3: Execute the program (type "readdata") and observe what happens.

CREATIVE WORK:

Add in more information into the data structure PersonalData.

Practice 3: To initialize a matrix with a constant and save the matrix to a file.

Step 1: Edit the following two files: initmatrix.c and Makefile.

```
#include <stdio.h>
#include <string.h>
typedef unsigned char   Byte ;
Byte      image[256][256] ;
main(int argc, char **argv)
{
        FILE *pf ;
        int     line, col ;
        Byte  value ;

        value = 255 ;
        for (line = 0; line < 256; line++)
        {
            for (col=0; col < 256; col++)
            {
                image[line][col] = value;
            }
         }
         pf = fopen("image.dat", "w") ;
         if (pf == NULL) exit(0) ;
         fwrite((char *) image, 1, 256*256, pf) ;
         fclose(pf) ;
}
```

initmatrix.c

```
.c.o:
        cc  -c   $*.c

initmatrix:   initmatrix.o
        cc -o    initmatrix   initmatrix.o
```

Makefile

Step 2: Compile the program (use "make initmatrix").

Step 3: Execute the program (type "initmatrix").

CREATIVE WORK:

Read in a matrix from a file and set the value of each element according to the following formulate:

$$image[i][j] = (i*j)/256$$

Practice 4: To reduce the dimension of a matrix by half and save it to a file.

Step 1: Edit the following two files: reducematrix.c and Makefile.

```
#include <stdio.h>
#include <string.h>
typedef unsigned char   Byte ;
Byte      image[256][256] ;
Byte     halfimage[128][128] ;
main(int argc, char **argv)
{
          FILE *pf ;
          int     line, col ;

          for (line = 0; line < 128; line++)
          {
               for (col=0; col < 128; col++)
               {
                    halfimage[line][col] =
                         image[2*line][2*col];
               }
           }
           pf = fopen("image.dat", "w") ;
           if (pf == NULL) exit(0) ;
           fwrite((char *) halfimage, 1, 128*128, pf) ;
           fclose(pf) ;
}
```

reducematrix.c

```
.c.o:
        cc  -c   $*.c

reducematrix:   reducematrix.o
        cc -o   reducematrix   reducematrix.o
```

Makefile

Step 2: Compile the program (use "make reducematrix").

Step 3: Execute the program (type "reducematrix").

CREATIVE WORK:

Read in a matrix from a file and reduce it by half. Save the resulting matrix to a new file.