Práctica 1

Recopilación de ejercicios básicos

Universidad Nacional de Rosario, Instituto Politécnico, Dto. de Informática,
prramis@ips.edu.ar,
WWW home page: http://informatica.ips.edu.ar

1. Matriz identidad

Escriba un programa que genere la matriz identidad de tamaño N x N.

2. Estadistica de notas

Un instituto desea controlar los resultados de los alumnos en las diferentes asignaturas de un curso de informática.

El programa debe leer las calificaciones obtenidas en las distintas asignaturas y visualizar en pantalla el número de cada estudiante seguido por su promedio, además se deberá imprimir al lado "libre", regular.º "promovido", si dicho promedio está en los intervalos [0,6), [6,8) y [8,10] respectivamente.

El programa visualizará también la calificación promedio de todos los estudiantes de cada asignatura.

Organice los datos de la siguiente manera:

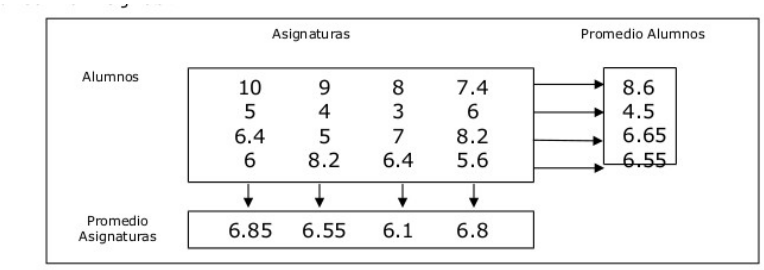

Figura 1. Datos del registro

3. Suma de matrices

Programa que realiza la suma de dos matrices de N x N.

La salida por pantalla tendría que ser similar a esta:

```
Teclea el ñtamao de la matriz: 4
Contenido de las matrices
Matriz 1
1 2 1 3
1 4 5 6
6 2 1 6
7 4 3 1
Matriz 2
2 3 2 5
5 4 2 1
8 7 6 9
9 5 4 1
Suma
3 5 3 8
6 8 8 7
14 9 7 15
16 9 7 2
```

4. Matriz Transpuesta

Escriba un programa que cree la transpuesta de una matriz.
La salida en pantalla tendría que ser similar a esta:

```
Teclea el únmero de renglones: 3
Teclea el únmero de columnas: 4
MATRIZ ORIGINAL
1 4 5 7
2 4 1 5
7 6 4 9
MATRIZ TRANSPUESTA
1 2 7
4 4 6
5 1 4
7 5 9
```

5. Descubriendo punteros

```
1
2  #incluce<stdio.h>
3
4  int main(){
5      int i = 8, *pi=&i;
6      long long l = 8, *pl=&l;
7      float f = 102.8f, *pf=&f;
8      double d=678.44, *pd=&d;
9      int vec[100];
10     vec[0] = 44;
11     printf("variable int, tam.bytes:   %d \tdir.&i:  %p \
           tvalor: %d\n", sizeof(i), &i, i);
12     printf("puntero int, tam.bytes=    %d \tdir.&pi: %p \
           tvalor: %p\n", sizeof(pi), &pi, pi);
```

```
13      printf("variable long, tam.bytes:  %d \tdir.&l:  %p \
            tvalor: %ld\n", sizeof(l), &l, l);
14      printf("puntero long, tam.bytes:   %d \tdir.&pl: %p \
            tvalor: %p\n", sizeof(pl), &pl, pl);
15      printf("variable float, tam.bytes: %d \tdir.&f:  %p \
            tvalor: %.1f\n", sizeof(f), &f, f);
16      printf("puntero float, tam.bytes:  %d \tdir.&pf: %p \
            tvalor: %p\n", sizeof(pf), &pf, pf);
17      printf("variable double, tam.bytes:%d \tdir.&d:  %p \
            tvalor: %.2lf\n", sizeof(d), &d, d);
18      printf("puntero double, tam.bytes: %d \tdir.&pd: %p \
            tvalor: %p\n", sizeof(pd), &pd, pd);
19      printf("variable array, tam.bytes: %d \tdir.&vec[0]: %p \
            tvalor: %d\n", sizeof(vec[0]), &vec[0], vec[0]);
20      printf("puntero array, tam.bytes:  %d \tdir.&vec: %p \
            tvalor: %p\n", sizeof(vec), &vec, vec);
21
22      return 0;
23  }
```

Verifique el tamaño de cada tipo de variable y del puntero asociado.

6. Escriba un programa que defina las siguientes variables

```
1
2   int i=5, j[]={1,2,3,4,5,6,7,8,9,10};
3   char x = 'a', pal [] ="texto en c";
4   int *pi;
5   char *pc;
```

1. Mostrar la dirección de "i" y su valor.
2. Mostrar los mismos valores a través del puntero "pi".
3. Recorrer el vector "j" mostrando para cada elemento, su dirección y valor.
4. Recorra el vector accediendo a través del puntero "pi" y usando álgebra de punteros.
5. Repita lo mismo con las variables char, el arreglo y el puntero.
6. Finalmente muestre la dirección donde se almacenan ambos punteros.

Genere una salida similar a esta:

```
Por Variable: 'i' Valor: 5 óDireccin: 13FF5C
Por Puntero: 'pi' Valor: 5 óDireccin: 13FF5C
Por Variable: 'j[0]' Valor: 1 óDireccin: 13FF2C
Por Puntero: ' pi(=&j)+0' Valor: 1 óDireccin: 13FF2C
Por Variable: 'j[1]' Valor: 2 óDireccin: 13FF30
Por Puntero: 'pi(=&j)+1' Valor: 2 óDireccin: 13FF30
...
```

```
Por Variable: 'x' Valor: a óDireccin: 13FF23
Por Puntero: 'pc' Valor: a óDireccin: 13FF23
Por Variable: 'pal[0]' Valor: t óDireccin: 13FF0C
Por Puntero: 'pc(=&pal)+0' Valor: t óDireccin: 13FF0C
Por Variable: 'pal[1]' Valor: e óDireccin: 13FF0D
Por Puntero: ' pc(=&pal)+1' Valor: e óDireccin: 13FF0D
...ó
Direccin de *pi: 13FF00 De *pc: 13FEF4
```

7. Recorriendo un array

Crear un programa que lea un número determinado (¡100) de reales introducidos por teclado, los almacene en un vector para luego mostrarlos en orden inverso. Para recorrer el array deberá usar aritmética de punteros en lugar de índices del array.

8. Retornar un puntero

Escribir una función que tome como argumento un entero positivo entre 1 y 7 y retorne un puntero a cadena con el nombre del día de la semana correspondiente al argumento. Probar dicha función.

9. Cadena de caracteres

Escribir las funciones que operan sobre cadenas de caracteres.

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  typedef enum { MAYUSCULAS, MINUSCULAS } may_min;
6
7  int strLargo(const char *origen); //Cantidad de caracteres
8
9  int strVacio(const char *origen); //retorna 1 si tiene al
       menos un caracter, 0 en otro caso
10
11 void strCopia(char *destino, const char *origen); // Copiador
12
13 /*prototipo modificado para permitir argumentos de tipo
       string literales, en casi todos los
14 compiladores un literal string es considerado una constante,
       o sea la ófuncin no ípodra
15 modificarlos pero, en algunos compiladores tales como GCC es
       posible modificarlos (úsegn
16 K&R el comportamiento es indefinido)*/
```

```
17  char* reverse(char *string);//retorna una cadena que es
        string invertida
18
19  void strIzq(char *destino, const char *origen); // Saca
        blancos Izq.
20
21  void strDer(char *destino, const char *origen); // Saca
        blancos Der.
22
23  void strAmbos(char *destino, const char *origen); // Saca
        blancos Izq. y Der.
24
25  void strMayMin(char *destino, const char *origen, may_min m);
         // Convierte May. Min.
26
27  int main(){
28      char *text1 =" Sera Cierto ?? ";
29      int largo=strLargo(text1)+1;
30      char *result = (char *)malloc (largo);
31      char* reves;
32      if(result == NULL)
33          return -1;//sino pudo reservar memoria para result
34      printf("La cadena: ");
35      puts(text1);
36      printf("Se encuentra: %s\n",(strVacio(text1) ? "No vacia"
            : "Vacia"));
37      printf("Largo : %d\n", strLargo(text1));
38      strCopia(result,text1);
39      printf("Copia : [%s]\n", result);
40      strIzq(result,text1);
41      printf("Sin blancos a la Izq:");
42      puts(result);
43      strDer(result,text1);
44      printf("Der : [%s]\n", result);
45      strAmbos(result,text1);
46      printf("Ambos: [%s], sin blancos al principio ni al final
            .\n", result);
47      strMayMin(result,text1, MAYUSCULAS);
48      printf("Mayusculas : [%s]\n", result);
49      strMayMin(result,text1, MINUSCULAS);
50      printf("Minusculas : [%s]\n", result);
51      reves=reverse(text1);
52      printf("La cadena: %s invertida queda: %s\n",text1, reves
            );
53
54      return 0;
55  }
```

La salida debería tener que ser similar a esta:

```
La cadena: Sera Cierto ??
Se encuentra: No vacia
Largo : 20
Copia : [ Sera Cierto ?? ]
Sin blancos a la Izq:Sera Cierto ??
Der : [ Sera Cierto ??]
Ambos: [Sera Cierto ??], sin blancos al principio ni al final.
Mayusculas : [ SERA CIERTO ?? ]
Minusculas : [ sera cierto ?? ]
La cadena: Sera Cierto ?? invertida queda ?? otreiC areS
Presione una tecla para continuar . . .
```