**Ejercicio No. 1**

El siguiente ejercicio muestra que los parámetros pasados por valor se pueden modificar dentro de

la función sin cambiar la variable que representa al parámetro en el llamador. Digite (póngale

extensión .cpp a su archivo), compile y ejecute el programa.

#include <iostream.h>

void DemoLocal( int valor )

{

cout << “Dentro de DemoLocal, valor =” << valor << endl;

valor = 75;

cout << “Dentro de DemoLocal, valor =” << valor << endl;

}

void main( void)

{

int n = 10;

cout << “Antes de llamar a DemoLocal, n= “ << n << endl;

DemoLocal(n);

cout << “Después de llamar a DemoLocal, n= “ << n << endl;

cout << “Pulse Intro para continuar”;

cin.get();

}

**¿Cómo cambia el valor de n, si es que lo hace?**

No cambia el valor de **n,** porque el paso de parámetros por valor le da una copia de los valores de los parámetros pasados como argumentos a la función **void DemoLocal** (**int** valor). Sin embargo aunque el valor copiado haya sido modificado dentro de la función **DemoLocal( )** nunca va a cambiar el valor original de **n.**

**Ejercicio No. 2**

El siguiente ejemplo muestra el uso de tres rutinas de intercambio de datos diferentes. Digite (póngale extensión .cpp a su archivo), compile y ejecute el programa.

#include <stdio.h>

#include <conio.h>

void Intercambio(int, int);

void Intercambio(int \*, int \*);

void Intercambio(int, int &, int &);

void main() {

int x = 22, y = 33;

printf("x = %d\t y = %d\n", x, y);

Intercambio(x,y); // llamada por valor

printf("x = %d\t y = %d\n", x, y);

PROGRAMACION II (SISTEMAS)

5

Intercambio(1, x, y); // llamada por referencia

printf("x = %d\t y = %d\n", x, y);

Intercambio(&x, &y); // llamada por puntero

printf("x = %d\t y = %d\n", x, y);

getch();

}

void Intercambio( int a, int b)

{

int aux;

aux = a;

a = b;

b = aux;

}

void Intercambio(int \*a, int \*b)

{

int aux;

aux = \*a;

\*a = \*b;

\*b = aux;

}

void Intercambio( int i, int &a, int &b)

{

int aux=i;

aux = a;

a = b;

b = aux;

}

* **Como resultado de la llamada a Intercambio por valor (la primera llamada de Intercambio()), ¿se cambian los valores de a y b?** NO **¿por qué?** Devido a que las variables reales no se pasan a la función, solo copias de su valor.Es decir, a pesar que la función reciba las copias de los valores de las variables a y b y que los valores de estas sean modificados dentro de la función nunca van a cambiar los valores originales de a y b (a = 22, b = 33).
* **Después de la segunda llamada a Intercambio() (llamada por referencia), ¿hubo modificación de a y b?** SI **¿por qué?** En este método el compilador no pasa una copia de los valores de a y b; en su lugar, pasa una referencia, que indica a la función dónde existen las variables en memoria. La referencia que la función recibe es la dirección de la variable y por ende le da acceso al valor de la variable original y al ejecutar la función modifica las valores de a y b (a = 33, b=22).
* **¿Es equivalente la tercer llamada a Intercambio() con la segunda llamada a la misma función?** NO **¿por qué?** Porque en la segunda llamada (referencia) se modificaron los valores originales de a y b (a = 33, b = 22), Y en la tercer llamada se utilizan punteros que permiten acceder a la dirección de las variables y lo cual hace posible cambiar los valores de a y b (a= 22, b =33)