**Erick Josué Saravia Aragon**

**SA18004**

**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?**

Una vez definida la variable “n” con el valor de 10, esta no cambia en ningún momento de la ejecución del programa

**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 &);

int 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);

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?** Porque los datos de las variables fueron copiados a otras que solo tienen alcance dentro de la función por lo que los cambios que estas sufran en ellas no tendrán efecto fuera de la ejecución de la función.

**Después de la segunda llamada a Intercambio() (llamada por referencia), ¿hubo modificación de a**

**y b?** Si **¿por qué?** Ya que los datos son guardados en el mismo espacio de memoria que las variables pasadas como parámetro en este caso “a” y “b”, al ser estas modificadas dentro de la funccion también lo hacen fuera de la misma.

**¿Es equivalente la tercer llamada a Intercambio() con la segunda llamada a la misma función?** Si

**¿por qué?**  La única diferencia entre ambas es que la tercera recibe un parámetro adicional que es obsoleto ya que se utiliza para dar valor a una variable, a la que se le cambia el valor inmediatamente. SI se suprime dicho parámetro en la función esta es equivalente a la segunda.