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>

using namespace std;

void DemoLocal( int valor )

{

cout << "Dentro de DemoLocal, valor =" << valor << endl;

valor = 75;

cout << "Dentro de DemoLocal, valor =" << valor << endl;

}

int 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? El valor de n se mantiene igual porque solo se hace el paso de variables por valor.

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);

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 en realidad solo se pasa una copia de los valores originales la cual es asignada a las variables internas de la función intercambio.

Después de la segunda llamada a Intercambio () (llamada por referencia), ¿hubo modificación de a y b? Si ¿por qué? Se está pasando por referencia, los cuales almacenan la posición de las variables (a y b) originales, de esta forma, cuando corre la función cambia el orden de las variables.

¿Es equivalente la tercera llamada a Intercambio () con la segunda llamada a la misma función? Si ¿por qué? Ambos funcionan de forma similar y cumplen la misma función con una sintaxis un tanto diferente, ambos almacenan la posición de la variable que se manda a llamar en la función, pero para llamar a ese valor en lo punteros se utiliza \*x, mientras que con referencia solo se pone la variable x. Por cómo funcionan los punteros se necesita acceder al valor de la variable a la que apunta (\*a), mientras que de la otra forma se utiliza una referencia al valor original (&a).

Ejercicios Propuestos:

1. Escribir una función **mayor()** que intercambie dos valores cuando el primero sea mayor que el segundo. Hacer un programa que la utilice.

2. Escribir una función que multiplique un valor por 10 y devuelva el valor modificado. Hacer un programa que la utilice.

3. Escribir una función **potencia()** , que calcule la potencia de un número n (tipo double) elevado a un exponente p (tipo int). Escribir un programa que haga uso de esta función.

4. Diseñar una función **area()** que pueda calcular el área de un cuadrado y un cubo. Escribir un programa que haga uso de esta función.

5. Escribir una función **menorAcero()** al que se pasan dos argumentos int por referencia y a continuación fijar el menor de los dos números a 0. Escribir un programa que utilice esta función.