Transformación de algoritmos recursivos a iterativos

Objetivos

- Realizar la transformación de algoritmos recursivos a iterativos utilizando el esquema general de transformación de algoritmos.
- Implementar algoritmos en C++.
- Gestionar pilas en un programa utilizando la librería stack.
- Trabajar en grupos de 2 o 3 miembros.

Actividades

Actividad 1

Dado el siguiente algoritmo

```
función calcular1(x:real, n:natural U {0}):real
   r:real
   si n=0
      r ← 1
   si no
      si impar(n)
         r ← x*calcular1(x,n-1)
      si no
         r ← calcular1(x,n/2)
         r ← r*r
      fsi
   fsi
   devolver r
ffunción
```

1.1.- Implementa en C++ la función correspondiente al algoritmo calcular1 aplicando las reglas de implementación de pseudocódigo que se vieron en la práctica 1 y crea un programa que pida por teclado dos números 'x' y 'n' y muestre el resultado de aplicar el algoritmo realizando una llamada a la función. En el caso de que 'n' sea un número negativo el programa imprimirá el texto Error.

1.2.- Genera el pseudocódigo de la versión iterativa del algoritmo aplicando el esquema general de transformación apropiado según el tipo de recursividad.

1.3.- Implementa en C++ la función correspondiente al algoritmo iterativo desarrollado en el apartado anterior y modifica el programa para incluir una llamada a esta función. Comprueba que se obtienen los mismos resultados que con el algoritmo recursivo.

Actividad 2

Dado el siguiente algoritmo

```
función calcular2(x:entero, y:entero):entero
   si x ≤ 4
      devolver x+y
   si no
      x ← x - 4
      y ← y / 3
      devolver calcular2(x, y) + x*y
   fsi
ffunción
```

2.1.- Implementa en C++ la función correspondiente al algoritmo calcular2 aplicando las reglas de implementación de pseudocódigo que se vieron en la práctica 1 y crea un programa que pida por teclado dos números 'x' e 'y' y muestre el resultado de aplicar el algoritmo realizando una llamada a la función.

2.2.- Genera el pseudocódigo de la versión iterativa del algoritmo aplicando el esquema general de transformación apropiado según el tipo de recursividad.

2.3.- Implementa en C++ la función correspondiente al algoritmo iterativo desarrollado en el apartado anterior y modifica el programa para incluir una llamada a esta función. Comprueba que se obtienen los mismos resultados que con el algoritmo recursivo.

Anexo: La clase pila en C++

En C++ existe una librería llamada stack que tiene implementada la clase pila y funciones para trabajar con ellas. Las funciones que incorpora esta clase son las siguientes:

Función	Descripción
push()	Introduce un elemento en la cima de la pila.
pop()	Elimina el elemento de la cima de la pila.
top()	Devuelve el elemento que se encuentra en la cima de la pila.
empty()	Verdadero si la pila está vacía. En caso contrario devuelve falso.
size()	Número de elementos de la pila.

Para declarar una variable del tipo pila se utiliza la instrucción

```
stack <tipodatos> nombrepila;
```

siendo tipodatos el tipo de datos de los datos que se guardan en la pila: int, float, double,…

En el siguiente ejemplo se muestra cómo utilizar una variable del tipo stack y la llamada a las funciones anteriores. Este programa declara una variable llamada ‘p’ del tipo stack y almacena en ella 5 números enteros que se introducen por teclado. A continuación, muestra el tamaño de la pila y finalmente imprime todos los elementos que se han apilado hasta dejar la pila vacía.

```
#include <iostream>
#include <stack>

using namespace std;

int main()
{
  stack <int> p;
  int dato,i;

  cout << "Apilando datos" << endl;
  cout << "--------------" << endl;
  for (i=1; i<=5; i++){
    cout << "Introduce dato: ";
    cin >> dato;
    p.push(dato);
  }

  cout << endl;
  cout << "Num. elementos de la pila: ";
  cout << p.size() << endl << endl;

  cout << "Desapilando datos" << endl;
  cout << "-----------------" << endl;
  while (! p.empty() )
  {
    dato = p.top();
    cout << dato << endl;
    p.pop();
  }
  cout << endl;
  system("pause");
  return 0;
}
```