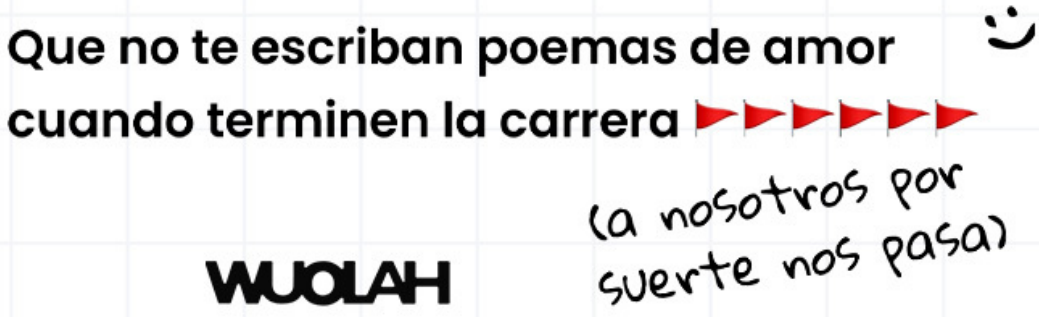
PRACTICA-5-RESUELTA-COMPLETA.pdf

user_2716437

Fundamentos de la Programación

1º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga



UNIVERSIDAD DE MÁLAGA
DPTO. DE LENGUAJES Y CC. DE LA COMPUTACIÓN
E.T.S. DE INGENIERÍA INFORMÁTICA

Práctica Nº 5
Recursividad

Ejercicios de clase.

1. Implementa una función recursiva que calcule la suma de los n primeros números naturales (sin contar el cero). La cabecera sería la siguiente:

   ```
   int sumaNaturales(int n)
   ```

   donde el parámetro `n` es el número de naturales a sumar. Diseña la función principal (main) para probar el funcionamiento de `sumaNaturales`. Para ello, se leerá de teclado el número de naturales a sumar (asegurándose que es un número > 0), se invocará a la función implementada y se mostrará por pantalla el valor devuelto por la misma. Por ejemplo, para una entrada de 5, la salida será 15.

2. Dados dos números naturales x y n, el valor de la potencia x^n se puede definir recursivamente del modo siguiente:

   $x^n=1$ si n=0

   $x^n = x*x^{n-1}$ si n>=1

   Implementa una función potencia que calcule recursivamente el valor de x^n con la siguiente cabecera:

   ```
   int potencia(int x, int n)
   ```

   Diseña la función principal (main) para probar el funcionamiento de `potencia`. Para ello, se leerán de teclado los valores x y n (asegurándose que son números naturales), se invocará a la función implementada y se mostrará por pantalla el valor devuelto por la misma. Por ejemplo, para una entrada de 5 para la x y de 3 para la y, la salida será 125.

3. Implementa una función recursiva que calcule el producto de dos números naturales x e y. La cabecera sería la siguiente:

   ```
   int producto(int x, int y)
   ```

   A la hora de diseñar la solución ten en cuenta que los únicos operadores aritméticos que puedes usar son la suma y la resta. Diseña la función principal (main) para probar el funcionamiento de `producto`. Para ello, se leerán de teclado los valores x e y (asegurándose que son números naturales), se invocará a la función implementada y se

mostrará por pantalla el valor devuelto por la misma. Por ejemplo, para una entrada de 4 para la x y de 5 para la y, la salida será 20.

4. Implementa un procedimiento recursivo que imprima los dígitos de un número natural n en orden inverso. Por ejemplo, para n=675 la salida debería ser 576. La cabecera sería la siguiente:

   ```
   void inverso (int n)
   ```

   Diseña la función principal (main) para probar el funcionamiento de `inverso`. Para ello, se leerá de teclado en valor de n (asegurándose que es un número natural) y se invocará al procedimiento implementado.

Ejercicio de refuerzo.

5. Implementa una función recursiva que devuelva true si el número natural que se le pasa como parámetro es primo y false en caso contrario. La cabecera de la función sería la siguiente:

   ```
   bool esPrimoRec (int num, int divisor)
   ```

   El primer parámetro es el número natural que se desea comprobar si es o no primo y el segundo parámetro es un valor para el que hay que comprobar si es o no divisor del primer parámetro. Inicialmente (en la primera llamada a `esPrimoRec`) el valor de ese segundo parámetro es 2.

   Diseña la función principal (main) para probar el funcionamiento de `esPrimoRec`. Para ello, se leerá de teclado el número natural (asegurándose que es un número > 1), se invocará a la función implementada y se mostrará por pantalla si el número es primo o no.

6. Implementa un procedimiento recursivo al que se le pase como parámetro un número natural n en base 10 (decimal), e imprima su valor en base 2 (binario). La cabecera sería la siguiente:

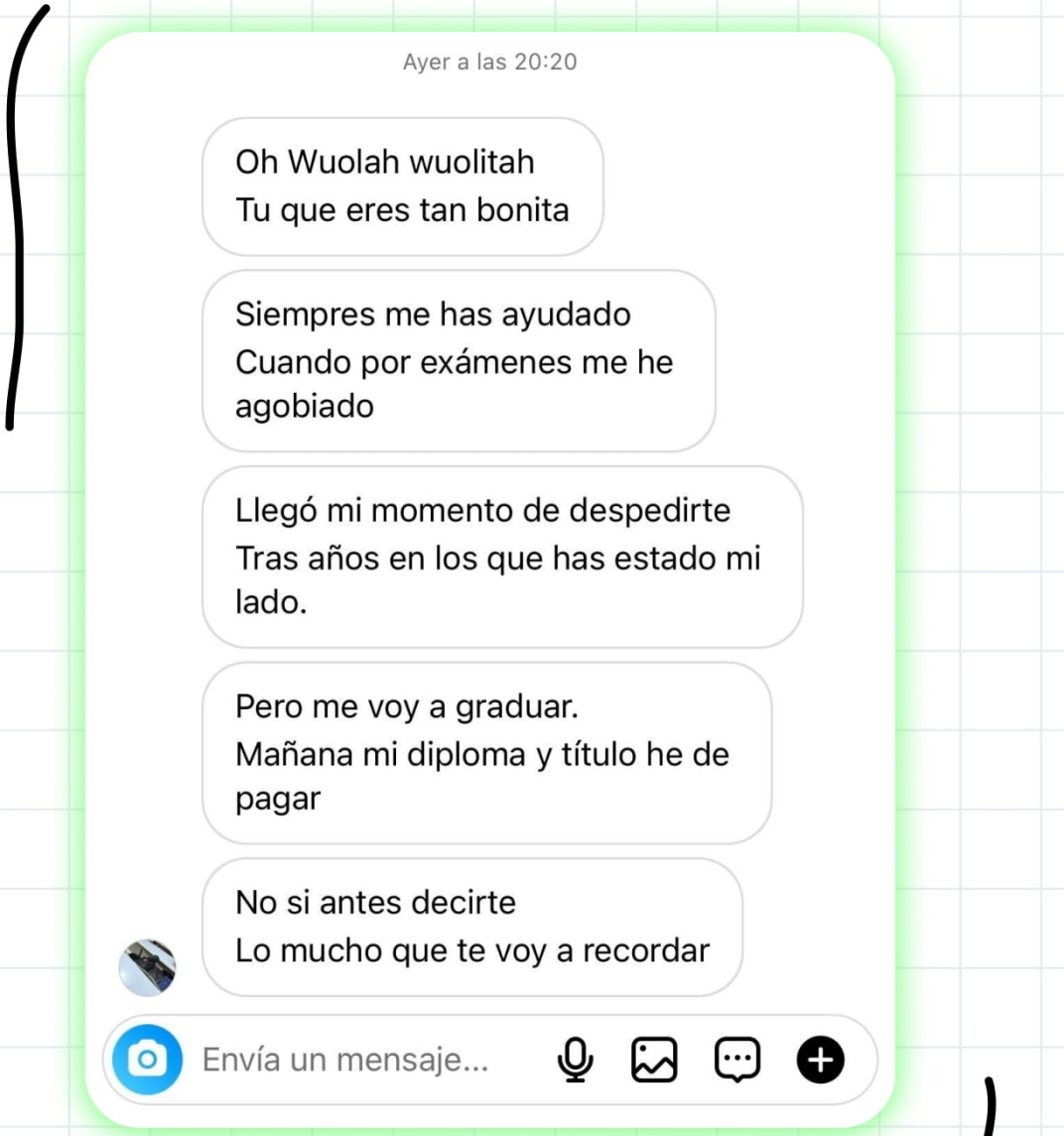
   ```
   void decimalAbinario (int n)
   ```

   Por ejemplo, para n=23, el procedimiento debería imprimir 10111.

   Diseña la función principal (main) para probar el funcionamiento de `decimalAbinario`. Para ello, se leerá de teclado en valor de n (asegurándose que es un número natural) y se invocará al procedimiento implementado.

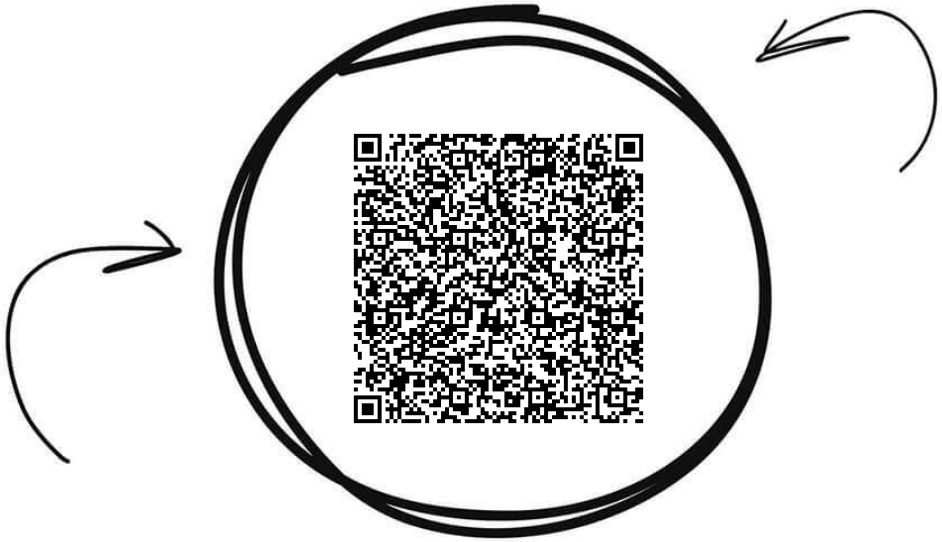
Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

Fundamentos de la Programación



Note bank of the

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

```
/*
Implementa una función recursiva que calcule la suma de los n primeros
números naturales
(sin contar el cero). La cabecera sería la siguiente:
int sumaNaturales(int n)
donde el parámetro n es el número de naturales a sumar. Diseña la función
principal (main)
para probar el funcionamiento de sumaNaturales. Para ello, se leerá de
teclado el número
de naturales a sumar (asegurándose que es un número > 0), se invocará a
la función
implementada y se mostrará por pantalla el valor devuelto por la misma.
Por ejemplo, para
una entrada de 5, la salida será 15.
*/

#include <iostream>
using namespace std;

int sumaNaturales(int &n);

int main(){
    int n = 0;
    while(n<=0){
        cout<<"Introduce el numero de naturales a sumar: ";
        cin>>n;
    }
    cout<<sumaNaturales(n);
    return 0;
}

int sumaNaturales(int &n){
    int sol = 0;
    for(int i=0; i<=n; i++){
        sol += i;
    }
    return sol;
}
```

```
/*
Dados dos números naturales x y n, el valor de la potencia x^n se puede
definir
recursivamente del modo siguiente:
x^n=1 si n=0
x^n=x*x^(n-1) si n>=1

Implementa una función potencia que calcule recursivamente el valor de
x^n con la siguiente cabecera:
int potencia(int x, int n)
Diseña la función principal (main) para probar el funcionamiento de
potencia. Para ello,
se leerán de teclado los valores x y n (asegurándose que son números
naturales), se invocará
a la función implementada y se mostrará por pantalla el valor devuelto
por la misma. Por
ejemplo, para una entrada de 5 para la x y de 3 para la y, la salida será
125.
*/

#include <iostream>
using namespace std;

int potencia(int &x, int &n);

int main(){
    int n = 0;
    int x = 0;
    while(n<=0 || x<=0){
        cout<<"Introduce x y n: ";
        cin>>x>>n;
    }
    cout<<potencia(x,n);
    return 0;
}

int potencia(int &x, int &n){
    int sol = 1;
    if(n=0){
        sol = 1;
    }else{
        for(int i=n; i>0; i--){
            sol *= x;
        }
    }
    return sol;
}
```

```
/*
Implementa una función recursiva que calcule el producto de dos números
naturales x e y.
La cabecera sería la siguiente:
int producto(int x, int y)

A la hora de diseñar la solución ten en cuenta que los únicos operadores
aritméticos que
puedes usar son la suma y la resta. Diseña la función principal (main)
para probar el
funcionamiento de producto. Para ello, se leerán de teclado los valores x
e y
(asegurándose que son números naturales), se invocará a la función
implementada y se
mostrará por pantalla el valor devuelto por la misma. Por ejemplo, para
una entrada de 4
para la x y de 5 para la y, la salida será 20.
*/

#include <iostream>
using namespace std;

int producto(int &x, int &y);

int main(){
    int y = 0;
    int x = 0;
    while(y<=0 || x<=0){
        cout<<"Introduce x y y: ";
        cin>>x>>y;
    }
    cout<<producto(x,y);
    return 0;
}

int producto(int &x, int &y){
    int sol = 0;
    for(int i=y; i>0; i--){
        sol +=x;
    }
    return sol;
}
```

```
/*
Implementa un procedimiento recursivo que imprima los dígitos de un
número natural n en
orden inverso. Por ejemplo, para n=675 la salida debería ser 576. La
cabecera sería la
siguiente:
void inverso (int n)

Diseña la función principal (main) para probar el funcionamiento de
inverso. Para ello, se
leerá de teclado en valor de n (asegurándose que es un número natural) y
se invocará al
procedimiento implementado.
*/

#include <iostream>
using namespace std;

void inverso (int n);

int main(){
    int n = 0;
    while(n<=0){
        cout<<"Introduce un numero: ";
        cin>>n;
    }
    inverso(n);
    return 0;
}

void inverso (int n){
    if(n<10){
        cout<<n;
    }else{
        cout<<n%10;
        inverso(n/10);
    }
}
```

```
/*
Implementa una función recursiva que devuelva true si el número natural
que se le pasa
como parámetro es primo y false en caso contrario. La cabecera de la
función sería la
siguiente:
bool esPrimoRec (int num, int divisor)

El primer parámetro es el número natural que se desea comprobar si es o
no primo y el
segundo parámetro es un valor para el que hay que comprobar si es o no
divisor del primer
parámetro. Inicialmente (en la primera llamada a esPrimoRec) el valor de
ese segundo
parámetro es 2.

Diseña la función principal (main) para probar el funcionamiento de
esPrimoRec. Para
ello, se leerá de teclado el número natural (asegurándose que es un
número > 1), se invocará
a la función implementada y se mostrará por pantalla si el número es
primo o no.
*/

#include <iostream>
using namespace std;

bool esPrimoRec (int num, int divisor);

int main(){
    int num = 1;
    int divisor = 2;
    while(num<=1){
        cout<<"Introduzca un numero: ";
        cin>>num;
    }
    if(esPrimoRec(num, divisor)){
        cout<<"Es primo";
    }else{
        cout<<"No es primo";
    }
    return 0;
}

bool esPrimoRec (int num, int divisor){
    while(divisor<num && num%divisor!=0){
        divisor++;
    }
    return divisor>=num;
}
```

```
/*
Implementa un procedimiento recursivo al que se le pase como parámetro un
número
natural n en base 10 (decimal), e imprima su valor en base 2 (binario).
La cabecera sería la
siguiente:
void decimalAbinario (int n)

Por ejemplo, para n=23, el procedimiento debería imprimir 10111.
Diseña la función principal (main) para probar el funcionamiento de
decimalAbinario.
Para ello, se leerá de teclado en valor de n (asegurándose que es un
número natural) y se
invocará al procedimiento implementado.
*/

#include <iostream>
using namespace std;

void decimalAbinario (int n);

int main(){
    int n = 0;
    while(n<=0){
        cout<<"Introduzca un numero: ";
        cin>>n;
    }
    decimalAbinario(n);
    return 0;
}

void decimalAbinario (int n){
    if(n/2!=0){
        decimalAbinario(n/2);
    }
    cout<<n%2;
}
```

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolitah
Tu que eres tan bonita

si lees esto me debes un besito

si lees esto me debes un besito

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolitah
Tu que eres tan bonita

si lees esto me debes un besito

si lees esto me debes un besito

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolitah
Tu que eres tan bonita

si lees esto me debes un besito

si lees esto me debes un besito

si lees esto me debes un besito