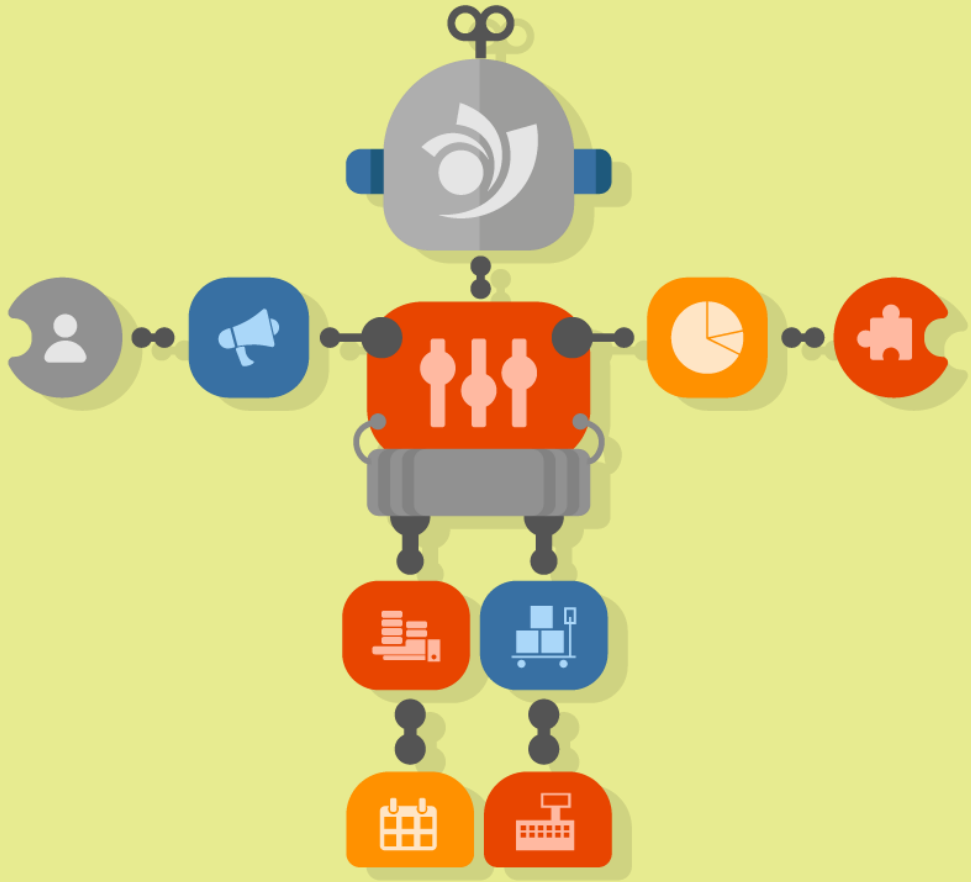
UNIVERSIDAD NACIONAL DE TRUJILLO

UNT

RECURSIVIDAD



Mg. Marcelino Torres Villanueva

¿QUE ES RECURSIVIDAD?

- La recursividad es un concepto fundamental en matemáticas y en computación.

- Es una alternativa diferente para implementar estructuras de repetición (ciclos). Los módulos se hacen llamadas recursivas.

- Se puede usar en toda situación en la cual la solución pueda ser expresada como una secuencia de movimientos, pasos o transformaciones gobernadas por un conjunto de reglas no ambiguas.

Divide y vencerás

La recursividad nos permite solucionar un problema bajo el criterio de “Divide y Vencerás”

Un problema complejo se divide en problemas más pequeños, de forma que se compone la solución al final a partir de las soluciones parciales que se van obteniendo.

Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema

FUNCION RECURSIVA

Las funciones recursivas se componen de:

- Caso base: Una solución simple para un caso particular (puede haber más de un caso base).
- Caso recursivo: Una solución que involucra volver a utilizar la función original, con parámetros que se acercan más al caso base. Los pasos que sigue el caso recursivo son los siguientes :
  1. La función se llama a sí misma.
  2. El problema se resuelve, resolviendo el mismo problema pero de tamaño menor
  3. La manera en la cual el tamaño del problema disminuye asegura que el caso base eventualmente se alcanzará

EJEMPLO : FACTORIAL

Muchas funciones matemáticas se definen recursivamente. Un ejemplo de ello es el factorial de un numero entero:

5! = 5 x 4 x 3 x 2 x 1
4! = 4 x 3 x 2 x 1
3! = 3 x 2 x 1
2! = 2 x 1
1! = 1 x 1
0! = 1

Sin recursividad

5! = 5 x 4!
4! = 4 x 3!
3! = 3 x 2!
2! = 2 x 1!
1! = 1 x 0!
0! = 1

Con recursividad

SOLUCION

Aquí podemos ver la secuencia que toma el factorial

$$N! = \begin{cases} 1 & \text{si } N = 0 \text{ (base)} \\ N * (N - 1)! & \text{si } N > 0 \text{ (recursión)} \end{cases}$$

Un razonamiento recursivo tiene dos partes: la base y la regla recursiva de construcción. La base no es recursiva y es el punto tanto de partida como de terminación de la definición.

factorial (5) = 5 * factorial(4) = 5 * 24 = 120

factorial (4) = 4 * factorial(3) = 4 * 6 = 24

factorial(3) = 3 * factorial(2) = 3 * 2 = 6

factorial(2) = 2 * factorial(1) = 2 * 1 = 2

factorial(1) = 1 * factorial(0) = 1 * 1 = 1

factorial(0) = 1 = 1

Ejercicio : Hacer un programa para calcular el factorial de un numero usando recursividad

```cpp
#include <iostream>
using namespace std;

int factorial(int n);

int main(int argc, char *argv[]) {
    int num;
    do{
        cout<<"Ingrese numero entero: ";
        cin>>num;
    } while(num<=0);
    cout<<"El factorial es : "<<factorial(num)<<endl;
    return 0;
}

int factorial(int n){
    if(n==0)
        return 1;
    else
        return n*factorial(n-1);
}
```

¿Recursividad o Iteración?

Semejanzas

- Ambas estrategias implican repetición: La iteración usa explícitamente una estructura de repetición que en la recursión esta implícita
- Ambas requieren una condición de corte: la iteración cuando llega al tope del contador y la recursión cuando alcanza el caso base.
- Ambas se aproximan a la solución gradualmente: la iteración modificando las variables iterativamente y la recursión produciendo versiones mas sencillas del problema original.
- Ambas pueden caer en bucles infinitos

Diferencias

- La recursividad produce algoritmos cortos y elegantes, mientras que la iteración presentan la habitual forma de uno o varios bucles.
- La recursión conlleva una repetida invocación de la función que, en general, incurre en un gasto de tiempo y de memoria que no se da en la versión iterativa.

EJERCICIOS RESUELTOS

01. Programa para calcular la potencia de x elevado a la n. x real y n>0

```
#include <iostream>
using namespace std;

float potencia(float x, int n);

int main(int argc, char *argv[])
{
    float x;
    int n;
    cout<<"Valor de x:";
    cin>>x;
    cout<<"valor de n :";
    cin>>n;
    cout<<"La Potencia es : "<<potencia(x,n)<<endl;
    return 0;
}

float potencia(float x, int n){
    if(n==0)
        return 1;
    else
        if(n<0)
            return 1/potencia(x,-n);
        else
            return x*potencia(x,n-1);
}
```

02. Crear una función recursiva para calcular el enésimo termino de la serie de Fibonacci

0 , 1, 1, 2, 3, 5, 8, 13, 21, …

Esta serie tiene la siguiente regla:

$$fibo(n) = \begin{cases} 0 & si\ n = 1 \\ 1 & si\ n = 2 \\ fibo(n-1) + fibo(n-2) & Si\ n > 3 \end{cases}$$

La función recursiva será:

```
16  int fibo(int n)
17  {
18      if(n==1) return 0;
19      else
20          if(n==2) return 1;
21          else
22              return fibo(n-1)+fibo(n-2);
23  }
```

```cpp
#include <iostream>
using namespace std;

int fibo(int n);

int main(int argc, char *argv[]) {
    int n;
    do{
       cout<<"Numero de Termino :";
       cin>>n;
    }while(n<=0);
    cout<<"El termino "<<n<<" es : "<<fibo(n)<<endl;
    return 0;
}

int fibo(int n)
{
    if(n==1) return 0;
    else
        if(n==2) return 1;
        else
            return fibo(n-1)+fibo(n-2);
}
```

03. Programa para calcular la suma de los dígitos de un numero

```
1   #include <iostream>
2   using namespace std;
3
4   int sumaDigitos(int n);
5   int main(int argc, char *argv[]) {
6       int num;
7       cout<<"Ingresar un numero : ";
8       cin>>num;
9       cout<<"La suma de los digitos es : "<<sumaDigitos(num)<<endl;
10      return 0;
11  }
12
13  int sumaDigitos(int n){
14      if(n==0)
15          return 0;
16      else
17          return n%10+ sumaDigitos(n/10);
18  }
```

04. Programa para ingresar un número y lo reporte al revés

```cpp
#include <iostream>
using namespace std;

void reporteReves(int n);

int main(int argc, char *argv[]) {
    int num;
    do{
        cout<<"Ingrese numero : ";
        cin>>num;
    }while(num<=0);
    reporteReves(num);
    return 0;
}

void reporteReves(int n)
{
    if(n>0)
    {
       cout<<n%10;
       reporteReves(n/10);
    }
}
```

05. Programa para ingresar un número y reporte los factores primos de un numero:

```cpp
#include <iostream>
using namespace std;

void factoresPrimos(int n, int d);

int main(int argc, char *argv[]) {
    int num;
    do{
        cout<<"Ingrese numero : ";
        cin>>num;
    }while(num<=0);
    factoresPrimos(num,2);
    return 0;
}

void factoresPrimos(int n, int d)
{
    if(n>1)
    {
        if (n % d ==0)
        {
            cout<<d<<" ";
            n=n/d;
            factoresPrimos(n,d);
        }
        else
            factoresPrimos(n,d+1);
    }
}
```

06. Programa para ingresar un número de base 10 y lo reporte a base b (entre 2 y 9)

```
#include <iostream>
using namespace std;

void reporteBaseb(int n,int b);

int main(int argc, char *argv[]) {

    int num,b;
    do{
        cout<<"Ingrese numero en base 10 : ";
        cin>>num;
    }while(num<=0);
    do{
        cout<<"Base a la que desea convertir : ";
        cin>>b;
    }while(b<2 || b>9);
    reporteBaseb(num,b);
    return 0;
}

void reporteBaseb(int n, int b)
{
    if (n>0)
    {
        reporteBaseb(n/b,b);
        cout<<n%b;
    }
}
```

07. Programa para calcular el máximo común divisor de 2 números usando el algoritmo de Euclides.

```cpp
#include <iostream>
using namespace std;
int mcd(int a, int b);

int main()
{
    int a,b;
    cout<<"Ingrese 2 numeros enteros:";
    cin>>a>>b;
    cout<<"El m.c.d. es : "<<mcd(a,b)<<endl;
    return 0;
}

int mcd(int a, int b){
    if(a % b==0)
        return b;
    else
        return mcd(b,a%b);
}
```