Universidad Autónoma del Estado de México

Centro Universitario UAEM Zumpango
Ingeniería en Computación

UA: Programación avanzada
Unidad de Competencia I

Tema: Recursividad

Dr. Asdrúbal López Chau

03/09/2018

Contenido

1. Propósito de la unidad de aprendizaje y de la UC
2. Guion explicativo de uso del material
3. Introducción
4. Funciones
5. Recursividad
6. Recursividad directa
7. Ejemplos de recursividad
8. Referencias

Propósito de la unidad de aprendizaje y de la UC

Propósito UA

Conocer, analizar y aplicar estructuras de datos estáticas y dinámicas mediante programas para la solución de problemas informáticos.

Propósito UC

Aplicar la estructura de datos árbol en el desarrollo de soluciones a problemas informáticos.

Conocimientos

Recursividad Directa: Definición. Funcionamiento.
Árboles: Usos. Características.
Árboles Binarios: Representación. Operaciones.
Recorrido (Preorden, Orden, Posorden)

Árboles Binarios de Expresiones: Características.
Evaluación de expresiones aritméticas.
Árboles Binarios de Búsqueda:Características.
Operaciones (Inserción, Eliminación, Búsqueda).

Guion explicativo de uso del material

En estas diapositivas se presenta el de recursividad. Se presentan algunos ejercicios para que los alumnos los realicen en clase, bajo la supervisión del docente.

Se anima a que los alumnos entiendan los ejemplos antes de implementarlos en algún lenguaje de programación. En las diapositivas se usa Java, lenguaje C y C++ sólo como demostración.

También se recomienda desarrollar algunos proyectos como actividad extra clase.

Introducción

En computación, la recursividad es una forma de resolver problemas en la cual un proceso es especificado en sí mismo, es decir, en su propia definición.

Otros nombres de la recursividad son Recurrencia o Recursión.

En programación, la recursión se implementa mediante funciones, por lo que comenzaremos este tema con un repaso de funciones.

Funciones

En programación, las funciones son bloques o módulos de código con un nombre que ejecutan una tarea específica.

Las funciones son usadas desde otras partes o áreas de código, mediante una llamada o invocación.

NOTA: Es importante enfatizar que una función debe de realizar sólo una tarea.

Funciones

Las funciones pueden aceptar datos como entradas, estos son usados de alguna forma para realizar operaciones dentro de la función, y luego esta última regresa algún resultado.

Los datos que son pasados a funciones en forma de argumentos.

Los argumentos pueden ser de diferente tipo, y la función puede regresar un sólo tipo (algunos lenguajes de programación permiten regresar más de un tipo).

NOTA: Hay que distinguir entre funciones y procedimientos. Aunque ambos realizan algún procesamiento, las funciones regresan un valor, mientras que los procedimientos no lo hacen.

Funciones

Es indispensable identificar la diferencia entre parámetro y argumento de una función.

Un parámetro es una variable en la definición de la función, es decir, en la declaración.

Un argumento es el dato (valor) que se pasa a los parámetros de una función.

Veamos un ejemplo.

Funciones

```
1  /*
2   * File:   main.c
3   * Author: Asdrúbal López Chau
4   *
5   */
6
7  #include <stdio.h>
8
9  int suma(int a, int b) {
10     return a + b;
11 }
12
13 int main() {
14     int a = 3;
15     int b = 4;
16     int resultado = suma(a, b);
17     return 0;
18 }
```

Un parámetro es una variable en la definición de la función, es decir, en la declaración.

parámetros

Declaración de la función **suma**

Argumentos

Invocación o llamado a la función **suma**

Un argumento es el dato (valor) que se pasa a los parámetros de una función.

Recursividad

Para resolver problemas la recursividad se necesita construir la solución en términos de sí misma.

Para algunas (la mayoría de las) personas, esta forma de atacar problemas resulta complicada al inicio, pero con un poco (mucha) práctica se puede adquirir destreza.

Recursividad

Existen dos tipos de recursividad:

Recursividad directa: Es la más común, y se implementa cuando una función se invoca a sí misma para resolver un problema.

Recursividad indirecta: Se implementa con más de una función. Por ejemplo, para dos funciones, la función A invoca la función B, y la B invoca a la A.

En este curso usaremos la recursividad directa.

Recursividad directa

En la recursividad directa, en alguna parte del cuerpo de la función se realiza una invocación a sí misma, usando como argumentos una versión más "pequeña" o más simple del problema.

También, en el cuerpo de la función debe de haber una forma de detener las llamadas recursivas, esto se logra a través del **caso base,** es decir, la forma directa de resolver la versión más simple del problema.

Ejemplos de recursividad directa

Primer ejemplo.

Comenzaremos con el clásico ejemplo: "Hola mundo". Imprimiremos N veces este mensaje de manera recursiva.

Para realizar la versión recursiva, tenemos que pensar en cuál sería el caso base...

Ejemplos de recursividad directa

Primer ejemplo

Imprimir “Hola mundo” N veces de manera recursiva.

El caso base puede ser cuando ya se haya impreso N veces el mensaje, es decir:

```
If se ha impreso N veces then
    Terminar
End
```

Ejemplos de recursividad directa

Primer ejemplo

Imprimir “Hola mundo” N veces de manera recursiva.

Si el problema (imprimir N veces) no ha sido resuelto por el caso base, entonces se debe de imprimir el mensaje e invocar recursivamente a la función.

Se debe hacer a que el nuevo problema que se atiende sea más simple (pequeño) cada vez.

Ejemplos de recursividad directa

Primer ejemplo

Imprimir “Hola mundo” N veces de manera recursiva.

Para lograr hacer que el problema es cada vez menor, usaremos un contador, que se decrementará en cada invocación. Esto nos permite deducir que la función recursiva debe tener un parámetro tipo entero (el contador).

La firma de la función puede ser así:

imprime(contador: entero)

Ejemplos de recursividad directa

Primer ejemplo

Imprimir “Hola mundo” N veces de manera recursiva.

Para determinar si se ha impreso N veces el mensaje, en el caso base se puede usar el contador:

```
If contador < 1 then
    Terminar
End
```

Ejemplos de recursividad directa

Primer ejemplo

Imprimir “Hola mundo” N veces de manera recursiva.

La solución recursiva es entonces:

```
imprimir(contador: entero)
    If contador < 1 then
        regresar
    End
    Print ”Hola mundo”
    imprimir(contador-1)
```

Ejemplos de recursividad directa

Una implementación en lenguaje C de esta función es la siguiente.

```c
/*
 * File:   main.c
 * Author: Asdrúbal López Chau
 */

#include <stdio.h>

void imprime(int contador) {
    if (contador < 1)return;//Caso base
    printf("Hola mundo\n");//
    imprime(contador-1);//Invocación recursiva
                        //Reduce el problema.
}

int main() {
    int N = 10;
    imprime(N);
    return 0;
}
```

Ejemplos de recursividad directa

Una implementación en lenguaje Java de esta función es la siguiente.

```java
package ejemplo113;

/*
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
CENTRO UNIVERSITARIO UAEM ZUMPANGO
INGENIERÍA EN COMPUTACIÓN
PROFESOR: ASDRÚBAL LÓPEZ CHAU
 */
/**
 *
 * @author asdruballopezchau
 */
public class HolaMundo {

    void imprime(int contador) {
        if (contador < 1) {
            return;
        }
        System.out.println("Hola mundo");
        imprime(contador - 1);

    }

    public static void main(String[] args) {
        HolaMundo hm = new HolaMundo();
        hm.imprime(10);

    }
}
```

Ejemplos de recursividad directa

Ejercicio.

Modificar la función anterior para que en el caso base sea como el siguiente

```
If contador > N then
    regresar
End
```

Ejemplos de recursividad directa

Ejemplo 2.

Calcular las siguiente sumatoria de manera recursiva

$$\sum_{i=1}^{n}(i * (i-1))$$

¿Cuál sería la forma del caso base?

Ejemplos de recursividad directa

Ejemplo 2. $\sum_{i=1}^{n}(i*(i-1))$

Para resolver este problema, el caso base es parecido al anterior ejemplo, es decir, se puede contar el número de veces que se ha realizado la suma. El caso base puede ser así:

```
If i > N then
    regresar
End
```

NOTA: La firma de la función debe incluir el parámetro i. Además, se puede de indicar el valor de N para saber hasta donde debe de llegar la sumatoria.

Ejemplos de recursividad directa

Ejemplo 2. $\sum_{i=1}^{n}(i*(i-1))$

Si no se ha llegado al caso base, entonces se debe de invocar a sí misma la función. Además, se debe de acumular el valor actual (i*(i-1)) con el resultado que regrese la siguiente invocación recursiva.

Sirve para determinar el límite de la sumatoria

Se suma uno para reducir el problema

```
If i > N then
    regresar
End
return i*(i-1) + sumatoria(N, i+1)
```

Ejemplos de recursividad directa

Ejemplo 2. $\sum_{i=1}^{n}(i*(i-1))$

Una solución recursiva de la sumatoria es la siguiente:

```
sumatoria(i,N)
    If i > N then
        regresar 0
    End
    return i*(i-1) + sumatoria(N,i+1)
```

Ejemplos de recursividad directa

Una implementación en lenguaje C++ se muestra aquí:

```
1  /*
2  UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
3  CENTRO UNIVERSITARIO UAEM ZUMPANGO
4  INGENIERÍA EN COMPUTACIÓN
5
6  DESCRIPCIÓN: \sum_{i=1}^{n}(i+1)(i-1)
7  PROFESOR: ASDRÚBAL LÓPEZ CHAU
8  */
9
10 #include<iostream>
11 using namespace std;
12
13 int sumatoria(int n, int i) {
14     if (i > n)return 0;//Caso base
15     return (i)*(i - 1) + sumatoria(n,i + 1);
16
17 }
18
19 int main(int argc, char const *argv[])
20 {
21     int numero = 0;
22     cout<<"Numero:"<<endl;
23     cin>>numero;
24     int suma = sumatoria(numero, 1);
25     cout<<"Sumatoria:"<<suma<<endl;
26     return 0;
27 }
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Aprovecharemos este ejemplo para ver dos versiones que resuelven este problema:

1. El primer enfoque es similar a los dos ejemplos anteriores.
2. El segundo enfoque usa una referencia para mantener el valor mayor.

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque I

La firma de la función requiere de los siguientes tres parámetros:

1. Arreglo (array)
2. Índice actual del arreglo (actual)
3. Tamaño del arreglo (N)

La función debe regresar el valor mayor

Nota: En algunos lenguajes (como Java) el mismo arreglo tiene un campo que contiene su tamaño

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque I

Si empezamos desde el índice menor del arreglo (cero), entonces necesitamos incrementarlo hasta el tamaño del arreglo. El caso base puede ser así:

Índice actual

Tamaño del arreglo

```
If actual > N then
    regresar el valor más pequeño posible
End
```

Para que no pueda ser el mayor buscado

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque I

Si el problema todavía no puede ser resuelto en el caso base, entonces se requiere determinar si el valor que contiene el índice actual del arreglo es mayor que el resultado que devuelva la siguiente llamada recursiva.

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque I

```
entero mayor(arreglo: enteros, actual: entero, N: entero)
    If actual > N then
            regresar el valor más pequeño posible
    End
    regresar máximo entre array[índice] y la siguiente llamada recursiva
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque I

```
entero mayor(arreglo: enteros, actual: entero, N: entero)
    If actual > N then
            regresar el
    End
    regresar máximo entre array[índice] y la siguiente llamada recursiva
```

Otra opción es usar el valor del PRIMER ELEMENTO DEL ARREGLO

Nota: Se puede usar el primer elemento del arreglo, pero usaremos un valor constante para mostrar la forma de obtener este valor en C/C++ y en Java.

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque I

La solución recursiva es entonces la siguiente:

```
entero mayor(arreglo: enteros, actual: entero, N: entero)
    If indice > N then
            return el valor más pequeño posible
    End
    return máximo entre array[índice] y la siguiente llamada recursiva
```

Valor mínimo de un entero de 32 bits

-2^{32-1}

mayor(arreglo, actual+1, N)

Ejemplos de recursividad directa

Implementación en C++

```
1  /*
2  UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
3  CENTRO UNIVERSITARIO UAEM ZUMPANGO
4  INGENIERÍA EN COMPUTACIÓN
5
6  PROFESOR: ASDRÚBAL LÓPEZ CHAU
7  */
8
9  #include<iostream>
10 using namespace std;
11 #include <ctype.h>
12
13 //Función para determinar el valor mayor de dos números
14 int max(int  a, int  b){
15     if (a > b)return a;
16     else return b;
17 }
18 //Función recursiva
19 double mayor(int* array,int tam, int actual) {
20     if (actual > tam - 1) {//Caso base
21         return -2^(32-1);//Valor mínimo de un entero
22     }
23     return max(mayor(array,tam, actual + 1),array[actual]);
24 }
```

Ejemplos de recursividad directa

Para probar la función recursiva:

```
25
26  int main(int argc, char const *argv[])
27  {
28      int *array = new int[N];
29      int max = 0;
30      for (int i = 0; i < N; ++i)
31      {
32          cout<<"Numero:"<<endl;
33          cin>>array[i];
34      }
35      max = mayor(array,N, 0);
36      cout<<"Mayor es: "<<max<<endl;
37      return 0;
38  }
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

En esta versión de la solución recursiva, la función no regresa ningún valor, en su lugar se mantiene una referencia hacia una variable que contiene el valor mayor.

Ejemplos de recursividad directa

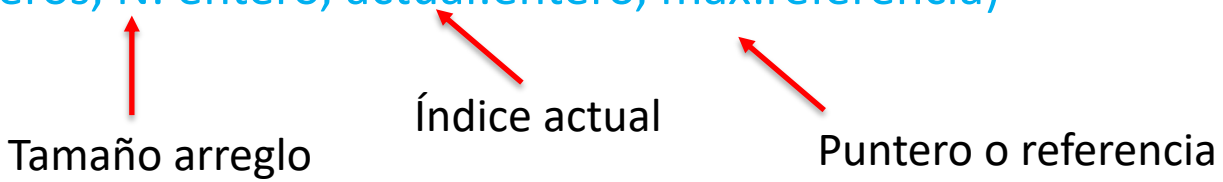
Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

La firma de la función puede ser así:

mayor(arreglo: enteros, N: entero, actual:entero, max:referencia)



Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

El caso base es similar a los anteriormente explicados.

```
If actual > N then
        regresar
End
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

Si no se ha alcanzado el caso base, entonces se debe de actualizar la variable max, con el valor del índice actual, si este es mayor, es decir:

```
If arreglo[actual] es mayor a  max Then
    Actualizar max a arreglo[actual]
End
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

Finalmente, se debe de continuar con la siguiente llamada recursiva:

```
mayor(array,N, actual + 1,max)
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

La solución completa es la siguiente:

```
mayor(arreglo: enteros, N: entero, actual:entero, max:referencia)
    If actual > N then
            regresar
    End
    If arreglo[actual] es mayor a  max Then
        Actualizar max a arreglo[actual]
    End
    mayor(array,N, actual + 1,max)
```

Ejemplos de recursividad directa

Ejemplo 3.

Dado un arreglo de enteros de longitud N, encontrar cuál es el valor más grande que contiene.

Enfoque II

Debido a que en algunos lenguajes no existen punteros, se mostrarán tres implementaciones

Implementación en lenguaje C
Implementación en lenguaje C++
Implementación en Java

Ejemplos de r

Implementación en lenguaje C

```c
/*
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
CENTRO UNIVERSITARIO UAEM ZUMPANGO
INGENIERÍA EN COMPUTACIÓN

PROFESOR: ASDRÚBAL LÓPEZ CHAU
*/

#include<stdio.h>
#include <ctype.h>
#define N 5
//Función recursiva para encontrar el mayor en un arreglo.
void mayor(int * array,int tam, int actual, int *max) {
    if (actual > tam - 1) {//caso base
        return;
    }
    if(array[actual]>*max)*max = array[actual];
    mayor(array,tam, actual + 1,max);
}

int main(int argc, char const *argv[])
{
    int array[N];
    int max = 1,i=0;
    for (i = 0; i < N; i++)
    {
        printf("Numero:\n");
        scanf("%d",&array[i]);
    }
    mayor(array,N,0,&max);
    printf("Mayor es: %d\n",max);
    return 0;
}
```

Ejemplos de re

Implementación en lenguaje C++

```cpp
/*
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
CENTRO UNIVERSITARIO UAEM ZUMPANGO
INGENIERÍA EN COMPUTACIÓN

PROFESOR: ASDRÚBAL LÓPEZ CHAU
*/

#include<iostream>
#include <ctype.h>
#define N 5
using namespace std;
//Función recursiva para encontrar el mayor en un arreglo.
void mayor(int * array,int tam, int actual, int &max) {
    if (actual > tam - 1) {//caso base
        return;
    }
    if(array[actual]>max)max = array[actual];
    mayor(array,tam, actual + 1,max);
}

int main(int argc, char const *argv[])
{
    int array[N];
    int max = 1,i=0;
    for (i = 0; i < N; i++)
    {
        cout<<"Numero:"<<endl;
        cin>>array[i];
    }
    mayor(array,N,0,max);
    cout<<"Mayor es:"<<max<<endl;
    return 0;
}
```

Ejemplos de re

Implementación en lenguaje Java

```java
import java.util.Scanner;
/*
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
CENTRO UNIVERSITARIO UAEM ZUMPANGO
INGENIERÍA EN COMPUTACIÓN
PROFESOR: ASDRÚBAL LÓPEZ CHAU
 * @author asdruballopezchau
 */
public class Recursivo01 {

    final int N = 5;
    int max = Integer.MIN_VALUE;

    //Función recursiva para encontrar el mayor en un arreglo.
    void mayor(Integer[] array, int tam, int actual) {
        if (actual > tam - 1) {//caso base
            return;
        }
        if (array[actual] > max) {
            max = array[actual];
        }
        mayor(array, tam, actual + 1);
    }

    public static void main(String[] args) {
        Recursivo01 hm = new Recursivo01();
        Scanner scanner = new Scanner(System.in);
        Integer[] array = new Integer[hm.N];
        for (int i = 0; i < hm.N; i++) {
            System.out.println("Numero:");
            array[i] = scanner.nextInt();
        }
        hm.mayor(array, hm.N, 0);
        System.out.println("El mayor es " + hm.max);

    }
}
```

En Java las variables fundamentales (int, long, double, etc) no se pasan por referencia. Hay que usar una variable de instancia.

Referencias

1. Cairó, Osvaldo y Guardati, Silvia. (2006). Estructuras de datos (3a. Edición) McGraw-Hill.
2. Joyanes, Luis. (2008). Fundamentos de programación (4a Edición). McGraw-Hill.
3. https://www.cs.utah.edu/~germain/PPS/Topics/functions.html Consultado el 10 de abril de 2018
4. López, Leobardo. (2004). Programación estructurada. Un enfoque algorítmico (2a. Edición). AlfaOmega.
5. Drozdeck, Adam. (2007). *Estructuras de datos y algoritmos en Java* (2a Edición). Thomson.

Próxima clase: árboles

UAEM

46