Recursión
Manejo de strings

Sofía Beatriz Pérez
Daniel Agustín Rosso

sperez@iua.edu.ar
drosso@iua.edu.ar

Centro Regional Univesitario Córdoba
Instituto Univeristario Areonáutico

Clase número 10 - Ciclo lectivo 2022

Agenda

Disclaimer

Los siguientes slides tienen el objetivo de dar soporte al dictado de la asignatura. De ninguna manera pueden sustituir los apuntes tomados en clases y/o la asistencia a las mismas.
Es importante mencionar que todos este material se encuentra en un proceso de mejora continua.
Si encuentra bugs, errores de ortografía o redacción, por favor repórtelo a sperez@iua.edu.ar y/o drosso@iua.edu.ar. También puede abrir issues en el repositorio de este link: ▸ infoI_IUA_GitLab

Introducción a la recursión I

Definición

Una función recursiva es una función que se llama a sí misma de manera directa o indirecta a través de otra función.

- La función recursiva en realidad sólo sabe cómo resolver el problema para el caso más sencillo, conocido como **caso base**.
- Si se invoca a la función desde el caso base, ésta simplemente devuelve un resultado.
- Si se llama a la función desde un problema más complejo, la función divide el problema en dos partes conceptuales. Una parte que la función sabe cómo resolver y una parte que la función no sabe cómo resolver.

Introducción a la recursión II

- La parte que la función no sabe resolver, es en general una versión mas simple del problema actual y hay un camino directo hasta llegar al caso base. Es por esto que una nueva llamada a la misma función, podría ser de ayuda para reducir aún más el problema.
- Debido a que este problema se parece al problema original, la función lanza (llama) a una nueva copia de sí misma para que trabaje con el problema más pequeño, a esto se le denomina llamada recursiva o también paso recursivo.

Ejemplos de algoritmos recursivos I

El factorial de un entero no negativo n, se escribe $n!$ y se calcula como $n * (n-1) * (n-2)$ donde por definición $1! = 1$ y $0! = 1$

```
1  #include<stdio.h>
2  int factorial(int n);
3  int main() {
4      int n=0,resultado;
5      printf("Ingrese un numero positivo\n");
6      scanf("%d",&n);
7      resultado=factorial(n);
8      printf("Factorial de %d = %d\n", n, resultado);
9      return 0;
10 }
11
12
13
```

Ejemplos de algoritmos recursivos II

```
14  int factorial(int n)
15  {
16      if (n>=1)
17          //Llamada recursiva
18          return n*  factorial(n-1);
19      else
20          //Caso base
21          return 1;
22  }
```

Ejemplos de algoritmos recursivos III

Calcular la suma de todos los números naturales comprendidos entre 0 y n usando recursión:

$$f(n) = 1 + 2 + 3 + ... + (n-2) + (n-1) + n \tag{1}$$

de forma genérica:

$$f(n) = \sum_{k=1}^{k=n} (k) \tag{2}$$

Ejemplos de algoritmos recursivos IV

```
1  int sum(int n)
2  {
3      int temp;
4
5      if (n==0)//Caso base
6      {
7          return(n)
8      }
9      else //Caso recursivo
10     {
11         temp = n + sum (n-1);
12         return (temp);
13     }
14 }
```

Ejemplos de algoritmos recursivos V

- Serie de Fibonacci
  - ▸ Ver en Wikipedia
- El problema del Templo de Benarés
  - ▸ Ver en Wikipedia
- Algoritmo de resolución de laberintos
  - ▸ Ver en Wikipedia

Definición recursiva

Una definición recursiva se utiliza para definir los elementos de un conjunto en términos de otros elementos del conjunto.

Ejemplos:

- Un número entero es 1 o $n+1$ donde n también es un número natural
- Reglas BNF[1] son sintaxis utilizadas para describir reglas gramaticales en lenguajes de programación[2].

$$< ex >::=< number > |(< ex > * < ex >)|(< ex > + < ex >)$$

$$(5 * ((3 * 6) + 8))$$

[1] Por sus siglas en inglés Backus normal form

[2] Esto se profundizará en DHS

Ventajas y desventajas

- Ventajas
  - Algoritmos expresados de forma más clara y comprensible
  - La codificación de un algoritmo en forma recursiva en general es mas compacta que de forma iterativa
- Desventajas
  - Consumo de memoria
  - Ejecuciones mas lentas
  - Pueden resultas mas complejos de entender

Cadenas terminadas en NULL

C++ soporta dos tipos diferentes de cadena de caracteres:

- Cadenas terminadas en NULL: es un arreglo de caracteres cuyo último caracter es el NULL.
- Objectos creados a partir de la clase string de C++

En muchas ocaciones a las cadenas del primer tipo se las conoce como "C-strings".

Cadenas terminadas en NULL: introducción

Cuando se declara un arreglo de caracteres para almacenar cadenas de texto, se debe reservar espacio para un caracter extra, por ejemplo para almacenar una palabra de 6 letras (DANIEL), se debe:

```
1 char my_name[7];
```

Es importante aclarar que no es necesario agregar el caracter de terminación NULL; el compilador realizará esto de forma automática.

```
1 char my_name[7]="DANIEL";
```

Cadenas terminadas en NULL: funciones I

C/C++ soportan una amplia gama de funciones para manipular y procesar este tipo de cadenas. Todas estan incluídas en la biblioteca string.h

- strcpy(s1,s2): copia el contenido de s2 en s1
- strcat(s1,s2): concatena s2 al final de s1
- strlen(s1): devuelve el largo de s1 (int)
- int strcmp (const char* str1, const char* str2): devuelve el largo de s1 (int)
  - 0: si ambas cadenas son iguales
  - 1: si el primer caracter que no coincide en $s1$ es mayor que en $s2$ es decir $s1 > s2$
  - 2: opuesto a s2

Cadenas terminadas en NULL: funciones II

- strchr(s1,ch): devuelve un puntero a la primer ocurrencia de ch en s1
- strchr(s1,s2): devuelve un puntero a la primer ocurrencia de s2 en s1. Si no existe, deuvelve NULL

Cadenas terminadas en NULL: funciones III

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main(void)
5  {
6      char s1[80], s2[80]="Hola",s3[80];
7      printf("Ingrese la primer cadena\n");
8      scanf("%s",s1);
9      printf("Longitud de s1 %ld\n",strlen(s1));
10     printf("Longitud de s2 %ld\n",strlen(s2));
11     printf("Ingrese la segunda cadena\n");
12     scanf("%s",s2);
13     printf("Longitud de s2 %ld\n",strlen(s2));
14     strcat(s1,s2);
15     printf("Longitud de s1 %ld\n",strlen(s1));
```

Cadenas terminadas en NULL: funciones IV

```
16      printf("%s\n",s1);
17      if(strcmp(s1,s3)==0)
18      {
19          printf("Son iguales\n");
20      }
21      else
22      {
23          printf("Son diferentes\n");
24      }
25      strcpy(s3,s1);
26
27
28
29
30
```

Cadenas terminadas en NULL: funciones V

```
31      if(strcmp(s1,s3)==0)
32      {
33          printf("Son iguales\n");
34      }
35      else
36      {
37          printf("Son diferentes\n");
38      }
39      return(0);
40  }
```

Cadenas en C++: introducción

C++ permite almacenar a una cadena como una secuencia de letras u otros caracteres:

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main(void)
6  {
7      string cadena;
8      cadena ="Ya estamos llegando al final de info II";
9      cout << cadena <<endl;
10     return(0);
11 }
```

Cadenas en C++: longitud de una cadena

La función lenght() devuelve la longitud de la cadena incluyendo espacios y signos de puntuación.

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main(void)
6  {
7      string cadena;
8      cadena ="Ya estamos llegando al final de info II";
9      cout << cadena.length()<<endl;
10     return(0);
11 }
```

Cadenas en C++: accediendo a caracteres

Se puede acceder a los caracteres utilizando el operador []. Las posiciones de los caracteres van desde 0 hasta $str.len() - 1$.

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  int main(void)
5  {
6      string cadena;
7      string caden2;
8      cadena ="Ya estamos llegando al final de info II";
9      cadena[0]='N';
10     cadena[1]='O';
11     cout << cadena<<endl;
12     return(0);
13 }
```

Cadenas en C++: asignación

Las cadenas en C++ estan diseñadas para tener un comportamiento similar a cualquier otro tipo de datos:

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  int main(void)
5  {    string cadena, cadena2;
6       cadena ="Ya estamos llegando al final de info II";
7       cadena[0]='N';
8       cadena[1]='O';
9       cadena2 = cadena;
10      cout << cadena2<<endl;
11      return(0);
12 }
```

Cadenas en C++: comparación

Se puede utilizar el operador == para comparar una cadena con otra. Devolverá verdadero si ambas cadenas son iguales:

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  int main(void)
5  {    string cadena;
6       cin>>cadena;
7       if(cadena=="daniel")
8            cout << "Son iguales";
9       else
10           cout << "Son diferentes";
11      return(0);
12 }
```

Cadenas en C++: concatenación I

Se puede utilizar el operador + para concatenar dos cadenas. También es posible utilizar += para contactenar una cadena al final de otra.

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4  int main(void)
5  {    string nombre, apellido, full;
6       nombre   = "Daniel";
7       apellido = "Rosso";
8       full = nombre + " " + apellido;
9       cout << full;
10      full = "";
```

Cadenas en C++: concatenación II

```
11        full += nombre;
12        full += " ";
13        full += apellido;
14        cout << full;
15        return(0);
16    }
```

¡Muchas gracias!

Consultas:

sperez@iua.edu.ar
drosso@iua.edu.ar