Algoritmos y programación

Dr. Omar Jorge Ibarra Rojas
[omar.ibarrarj@uanl.edu.mx]

Centro de Investigación en Ciencias Físico-Matemáticas
Universidad Autónoma de Nuevo León

febrero-junio 2021

Agenda

Preliminares de algoritmos y programación

Introducción

Algoritmo: "métodos para resolver problemas, los cuales son apropiados para implementarse computacionalmente" (Sedgewic, 1983).

Es necesario formalizar los algoritmos en sus representaciones más comunes: lista de pasos, diagrama de flujo y pseudo-código.

Introducción

Algoritmo: "métodos para resolver problemas, los cuales son apropiados para implementarse computacionalmente" (Sedgewic, 1983).

Es necesario formalizar los algoritmos en sus representaciones más comunes: lista de pasos, diagrama de flujo y pseudo-código.

¿Algún ejemplo?

Introducción

Las más simples ideas a ser implementadas en una técnica de computo científico, requieren una traducción a lenguaje máquina.

Figure: Theodore Twombly hablando con su computadora (ya quisiéramos).

Introducción

Lenguaje de programación: es el idioma utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Se usa entre otras cosas: constantes, variables, arreglos, operadores aritméticos, funciones, condiciones lógicas, etc.

Ejercicios vol. 1

- Formula y estructura condicional `if`:
  1. Calcule la longitud de una circunferencia de radio conocido.
  2. Calcule el perímetro de un rectángulo.
  3. Diga si un número es par o impar.
  4. Obtenga el valor absoluto de un número.
  5. Si dos números son positivos, calcule su producto, en caso contrario calcule su suma.
- Estructura repetitiva `for` o `while`:
  1. Para 5 números dados, determine el mínimo y el máximo.
  2. Dados 5 números y diga si están ordenados crecientemente.
  3. Dados 3 números diferentes e indique cual es el valor intermedio.
  4. Evaluar una función $f(x)$ por tramos.
  5. Muestre todos los pares entre a y b, y diga cuántos son.
  6. De n números, identifique el mínimo, máximo y promedio.
  7. Determine cuántos dígitos tiene un número entero dado.

Ejercicios vol. 2: arreglos y vectores

1. Dado un vector de datos invierta las posiciones de sus datos.
2. Encuentre el mayor, el menor y el promedio de un conjunto de datos de un vector.
3. Dado un vector de n enteros y calcular la media de los que estén en posiciones pares.
4. Hacer el producto de un vector por un escalar.
5. Hacer el producto punto de dos vectores.
6. Sumar y multiplicar dos matrices.
7. Ordenar un vector decrecientemente.

Nota: Complejidad de algoritmos

Introducción a C++

Lenguaje C++

C++ es un lenguaje de alto nivel orientado a **objetos** con una diversidad de aplicaciones. Es principalmente fuerte, en el computo científico.

Lenguaje C++

Estructura:

```
1 // my second program in C++
2 #include <iostream>
3 using namespace std;
4
5 int main ()
6 {
7   cout << "Hello World! ";
8   cout << "I'm a C++ program";
9 }
```

```
Hello World! I'm a C++ program
```

Lenguaje C++

Tipos de datos:

Keyword	*Variable Type*	*Range*
`char`	Character (or string)	-128 to 127
`int`	Integer	-32,768 to 32,767
`short` `short int`	Short integer	-32,768 to 32,767
`long`	Long integer	-2,147,483,648 to 2,147,483,647
`unsigned char`	Unsigned character	0 to 255
`unsigned int`	Unsigned integer	0 to 65,535
`unsigned short`	Unsigned short integer	0 to 65,535
`unsigned long`	Unsigned long integer	0 to 4,294,967,295
`float`	Single-precision floating point (accurate to 7 digits)	$\pm 3.4 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$
`double`	Double-precision floating point (accurate to 15 digits)	$\pm 1.7 \times 10^{-308}$ to $\pm 1.7 \times 10^{308}$

Lenguaje C++

Declaración de variables:

```
1  // operating with variables
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    // declaring variables:
9    int a, b;
10   int result;
11
12   // process:
13   a = 5;
14   b = 2;
15   a = a + 1;
16   result = a - b;
17
18   // print out the result:
19   cout << result;
20
21   // terminate the program:
22   return 0;
23 }
```

```
4
```

Lenguaje C++

Operadores de asignación compuestos:

expression	equivalent to...
`y += x;`	`y = y + x;`
`x -= 5;`	`x = x - 5;`
`x /= y;`	`x = x / y;`
`price *= units + 1;`	`price = price * (units+1);`

and the same for all other compound assignment operators. For example:

```
1  // compound assignment operators
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int a, b=3;
8    a = b;
9    a+=2;              // equivalent to a=a+2
10   cout << a;
11 }
```

```
5
```

Lenguaje C++

Operadores de comparación:

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Here there are some examples:

```
1 (7 == 5)     // evaluates to false
2 (5 > 4)      // evaluates to true
3 (3 != 2)     // evaluates to true
4 (6 >= 6)     // evaluates to true
5 (5 < 5)      // evaluates to false
```

Lenguaje C++

Operadores lógicos (!, &&, ||):

```
1 !(5 == 5)    // evaluates to false because the expression at its righ
2 !(6 <= 4)    // evaluates to true because (6 <= 4) would be false
3 !true        // evaluates to false
4 !false       // evaluates to true
```

```
1 ( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )
2 ( (5 == 5) || (3 > 6) )  // evaluates to true ( true || false )
```

Lenguaje C++

Estructura lógica `If-else`:

For example:

```
1 if (x == 100)
2   cout << "x is 100";
3 else
4   cout << "x is not 100";
```

This prints `x is 100`, if indeed x has a value of 100, but if it does not, and only if it does not, it prints `x is not 1` instead.
Several if + else structures can be concatenated with the intention of checking a range of values. For example:

```
1 if (x > 0)
2   cout << "x is positive";
3 else if (x < 0)
4   cout << "x is negative";
5 else
6   cout << "x is 0";
```

Lenguaje C++

Estructura repetitiva `while`:

```
1  // custom countdown using while
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int n = 10;
8
9    while (n>0) {
10     cout << n << ", ";
11     --n;
12   }
13
14   cout << "liftoff!\n";
15 }
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

Lenguaje C++

Estructura repetitiva `for`:

```
1  // countdown using a for loop
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    for (int n=10; n>0; n--) {
8      cout << n << ", ";
9    }
10   cout << "liftoff!\n";
11 }
```

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

Lenguaje C++

Arreglos: Herramienta para concatenar datos del mismo tipo.

```
1  // arrays example
2  #include <iostream>
3  using namespace std;
4
5  int foo [] = {16, 2, 77, 40, 12071};
6  int n, result=0;
7
8  int main ()
9  {
10   for ( n=0 ; n<5 ; ++n )
11   {
12     result += foo[n];
13   }
14   cout << result;
15   return 0;
16 }
```

```
12206
```

Lenguaje C++

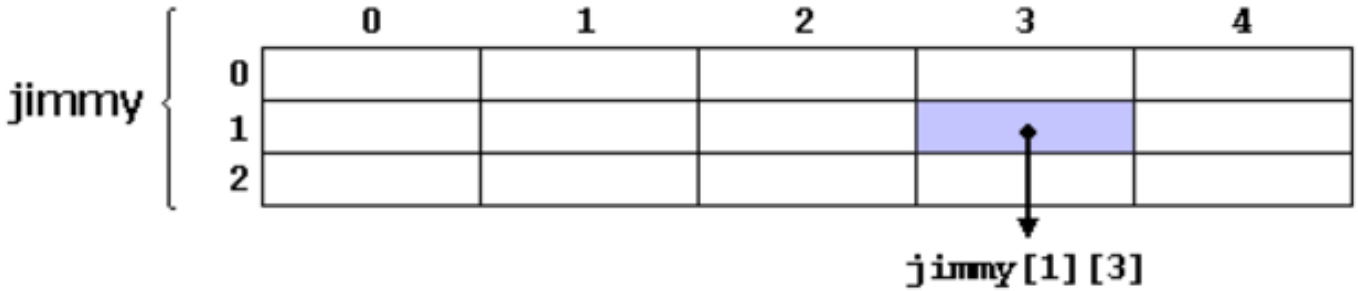
Arreglos:

`jimmy` represents a bidimensional array of 3 per 5 elements of type `int`. The C++ syntax for this is

```
int jimmy [3][5];
```

and, for example, the way to reference the second element vertically and fourth horizontally in an e

```
jimmy[1][3]
```



Lenguaje C++: Funciones

Una función es un conjunto de sentencias de control que cuentan con un nombre y puede llamarse desde algún punto de un programa.

Lenguaje C++: Funciones

Ejemplo de función suma:

```
1  // function example
2  #include <iostream>
3  using namespace std;
4
5  int addition (int a, int b)
6  {
7    int r;
8    r=a+b;
9    return r;
10 }
11
12 int main ()
13 {
14   int z;
15   z = addition (5,3);
16   cout << "The result is " << z;
17 }
```

```
The result is 8
```

Lenguaje C++: Funciones

Ejemplo de función factorial:

```
1  // factorial calculator
2  #include <iostream>
3  using namespace std;
4
5  long factorial (long a)
6  {
7    if (a > 1)
8     return (a * factorial (a-1));
9    else
10    return 1;
11 }
12
13 int main ()
14 {
15   long number = 9;
16   cout << number << "! = " << factorial (number);
17   return 0;
18 }
```

```
9! = 362880
```

Ejercicios vol. 3

Codifique un programa que

1. Reciba dos números como argumento y determine el mínimo.

2. Reciba como argumento un arreglo y un entero representando el tamaño de dicho arreglo para calcular e imprimir la media y la varianza.

Lenguaje C++

Estructura de datos: "arreglos" para diferentes tipos de datos.

```
struct type_name {
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
.
.
} object_names;
```

```
1 struct product {
2   int weight;
3   double price;
4 } ;
5
6 product apple;
7 product banana, melon;
```

Lenguaje C++

Para accesar a los atributos de las estructuras “producto”:

```
1 apple.weight
2 apple.price
3 banana.weight
4 banana.price
5 melon.weight
6 melon.price
```

Lenguaje C++

Para accesar a los atributos de las estructuras "producto":

```
1 apple.weight
2 apple.price
3 banana.weight
4 banana.price
5 melon.weight
6 melon.price
```

NOTA: Una vez definida, una estructura se trata como un tipo de dato, pudiendo así crear arreglos de ellas, concatenar estructuras, etc.

Ejercicios vol. 3

Crear la lista del grupo del curso de “programación y algoritmos” en un arreglo de estructuras “Alumno” con los siguientes atributos.

- Nombre
- Edad
- Matrícula

Imprimir la información del grupo con una estructura repetitiva.

Lenguaje C++: Clases

Una clase define un nuevo tipo de dato que especifica la forma de un objeto. Una clase incluye los datos y el código que operará sobre esos datos.

Ejemplo de clase "*rectangle*":

```
1 class Rectangle {
2     int width, height;
3   public:
4     void set_values (int,int);
5     int area (void);
6 } rect;
```

Lenguaje C++: Clases

Definición y uso de clase "*rectangle*":

```
1  // classes example
2  #include <iostream>
3  using namespace std;
4
5  class Rectangle {
6      int width, height;
7    public:
8      void set_values (int,int);
9      int area() {return width*height;}
10 };
11
12 void Rectangle::set_values (int x, int y) {
13   width = x;
14   height = y;
15 }
16
17 int main () {
18   Rectangle rect;
19   rect.set_values (3,4);
20   cout << "area: " << rect.area();
21   return 0;
22 }
```

```
area: 12
```

Lenguaje C++: Clase lista

Una lista de datos o estructuras. Atributos de la clase lista:

Capacity:

empty	Test whether container is empty (public member function)
size	Return size (public member function)
max_size	Return maximum size (public member function)

Element access:

front	Access first element (public member function)
back	Access last element (public member function)

Modifiers:

assign	Assign new content to container (public member function)
emplace_front C++11	Construct and insert element at beginning (public member function)
push_front	Insert element at beginning (public member function)
pop_front	Delete first element (public member function)
emplace_back C++11	Construct and insert element at the end (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
emplace C++11	Construct and insert element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
resize	Change size (public member function)
clear	Clear content (public member function)

Lenguaje C++: Clase lista

Ejemplo de declaración y algunas funciones:

```cpp
#include <iostream>
#include <list>

using namespace std;

int main(){

    //declarando clase del tipo lista
    list <int> L;

    //Insertando el número 8 al inicio de la lista
    L.push_front(8);

    //Definiendo iterador para explorar la lista
    list<int>::iterator it;

    //Ubicar el iterador al inicio de la lista
    it = L.begin();

    //Imprimir tamaño de la lista y el primer elemento
    cout << "El tamano de la lista L es: " << L.size() << endl;
    cout << "El primer elemento es: " << *it << endl;


    cout << "FIN DE PROGRAMA" << endl;
    return 0;
}
```

Lenguaje C++: Clase lista

Ejemplo de declaración y algunas funciones:

```
1  // list::push_back
2  #include <iostream>
3  #include <list>
4
5  int main ()
6  {
7    std::list<int> mylist;
8    int myint;
9
10   std::cout << "Please enter some integers (enter 0 to end):\n";
11
12   do {
13     std::cin >> myint;
14     mylist.push_back (myint);
15   } while (myint);
16
17   std::cout << "mylist stores " << mylist.size() << " numbers.\n";
18
19   return 0;
20 }
```

Lenguaje C++: Clase vector

Un vector es un arreglo multidimensional pero con la particularidad de que es de dimensión dinámica.

Lenguaje C++: Clase vector

Atributos de clase vector:

Capacity:

size	Return size (public member function)
max_size	Return maximum size (public member function)
resize	Change size (public member function)
capacity	Return size of allocated storage capacity (public member function)
empty	Test whether vector is empty (public member function)
reserve	Request a change in capacity (public member function)
shrink_to_fit C++11	Shrink to fit (public member function)

Element access:

operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
data C++11	Access data (public member function)

Modifiers:

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)

Lenguaje C++: Clase vector

Ejemplo de uso de clase vector:

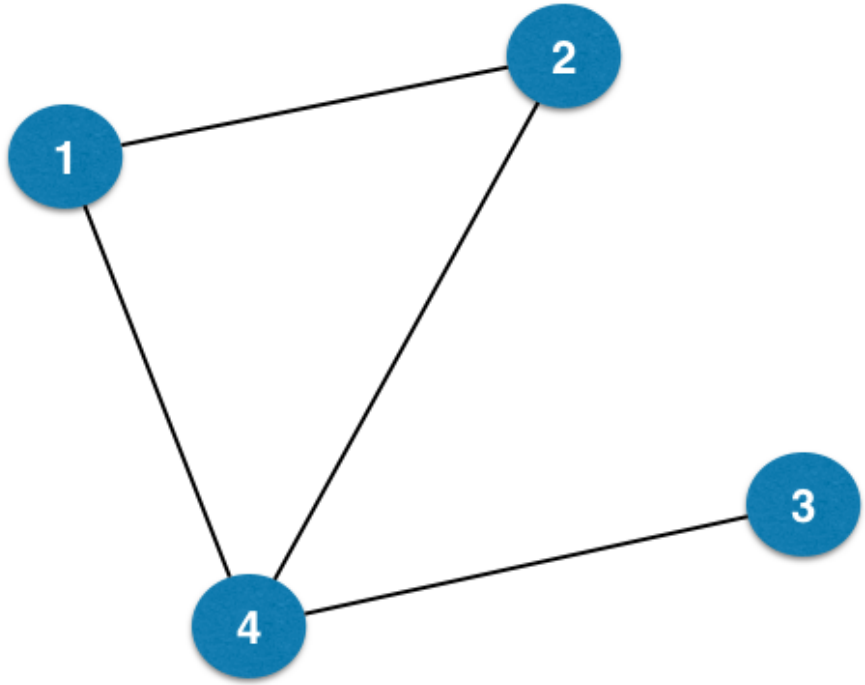
```
8  #include <iostream>
9  #include <vector>
10
11 using namespace std;
12
13 int main(){
14
15     //declarando clase del tipo lista
16     vector <int> V;
17
18     //dimensionando vector
19     V.resize(3);
20
21     //Agregando elementos al vector
22     V[0] = 10;
23     V[1] = 20;
24     V[2] = 30;
25
26     //Eliminando el elemento en la posición 1
27     V.erase( V.begin() + 1);
28     cout << V[0] << "," << V[1] << endl;
29
30     return 0;
31 }
```

Lenguaje C++: Clase vector

Ejemplo de uso de clase vector:

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main(){

    //declarando clase del tipo lista
    vector <vector <int> > V;

    //dimensionando vector
    V.resize(3);

    //Redimensionando vectores
    V[0].resize(2);
    V[1].resize(3);
    V[2].resize(4);

    //Redimensionando vectores otra vez
    for(int i = 0; i < 3; i ++){
        V[i].resize(3);
    }

    //Definiendo elemendo de fila 2 y columna 3
    V[1][2] = 2;

    return 0;
}
```

Ejemplo introductorio: Representación de un grafo



Ejercicio:

Dado un grafo como una matriz de adyacencia por nodos, generar un código que construya la representación de lista de adyacencia por nodos.

Ejercicio:

Dadas matrices $A_{m\times n}$ y $B_{n\times l}$ como vector de vectores, generar un código que identifique la cantidad de filas y columnas de cada matriz, además de lo siguiente:

1. La suma de los elementos de cada fila $i = 1, \ldots, m$ (resp, para B).
2. La suma de los elementos de cada columna $j = 1 \ldots, n$ (resp, para B).
3. La matriz transpuesta de A y B.
4. La cantidad de elementos diferentes en la matriz A.
5. El producto $C = AB$.