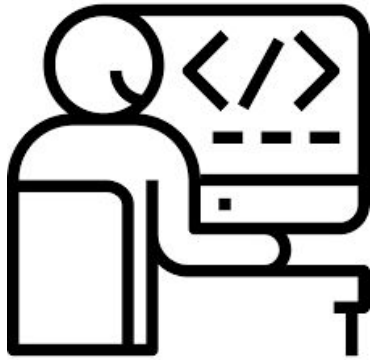


Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Corrección de Programas

CADP – CORRECCION DE PROGRAMAS

Cuando se desarrollan los algoritmos hay dos conceptos importantes que se deben tener en cuenta: CORRECCIÓN y EFICIENCIA del programa.

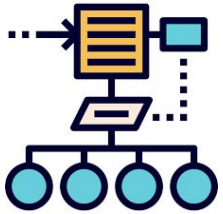
Un programa es correcto si se realiza de acuerdo a sus especificaciones.

Técnicas para corrección de programas

CADP – CORRECCION DE PROGRAMAS

TESTING

El propósito del Testing es proveer evidencias convincentes que el programa hace el trabajo esperado.

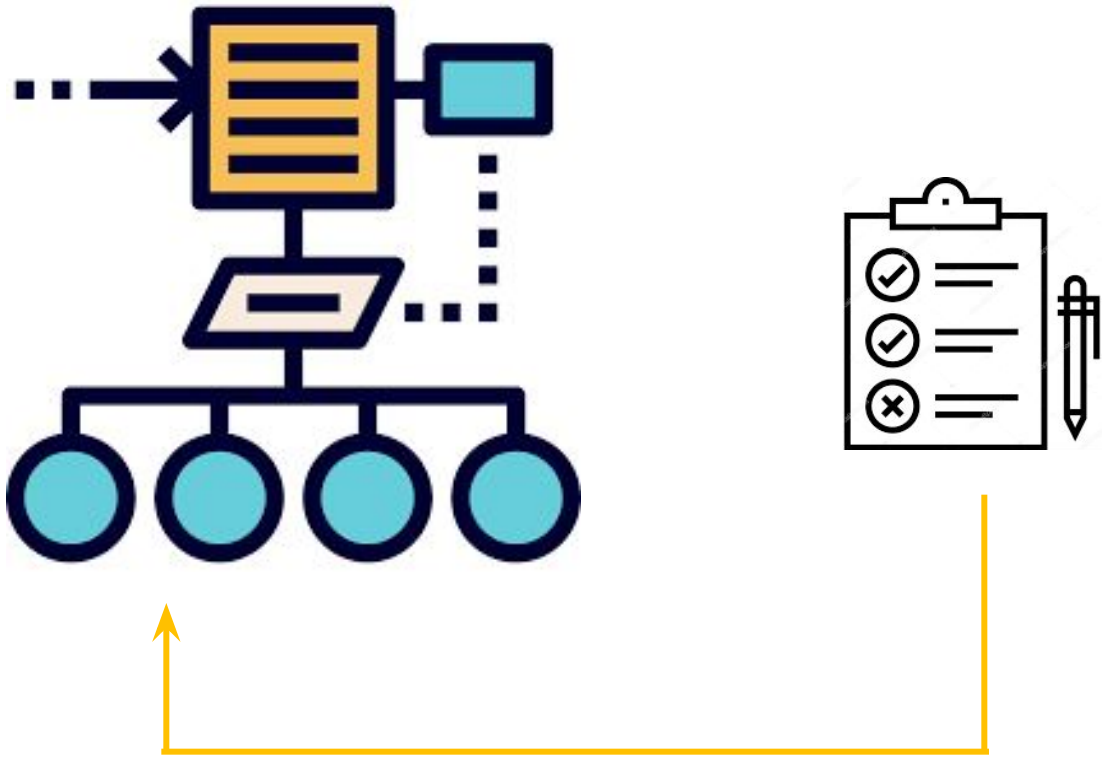


Diseñar un plan de pruebas

- Decidir cuales aspectos del programa deben ser testeados y encontrar datos de prueba para cada uno de esos aspectos.
- Determinar el resultado que se espera que el programa produzca para cada caso de prueba.
- Poner atención en los casos límite.
- Diseñar casos de prueba sobre la base de lo que hace el programa y no de lo que se escribió del programa. Lo mejor es hacerlo antes de escribir el programa.

CADP – CORRECCION DE PROGRAMAS TESTING

Una vez que el programa ha sido implementado y se tiene el plan de pruebas:



- Se analiza el programa con los casos de prueba.
- Si hay errores se corrigen.

Estos dos pasos se repiten hasta que no haya errores.

CADP – CORRECCION DE PROGRAMAS

DEBIGGING

Es el proceso de descubrir y reparar la causa del error.

Para esto pueden agregarse sentencias adicionales en el programa que permiten monitorear el comportamiento más cercanamente.

Los **errores** encontrados pueden ser de **tres tipos**

- SINTACTICOS: Se detectan el la compilación
- LOGICOS: Generalmente se detectan en la ejecución
- DE SISTEMA: Son muy raros los casos en los que ocurren

CADP – CORRECCION DE PROGRAMAS WALKTHROUGHS

Es el proceso de recorrer un programa frente a una audiencia.

La lectura de un programa a alguna otra persona provee un buen medio para detectar errores.

→ Esta persona no comparte preconceptos y está predispuesta a descubrir errores u omisiones.

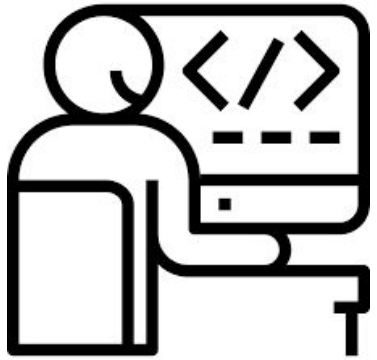
→ A menudo, cuando no se puede detectar un error, el programador trata de probar que no existe, pero mientras lo hace, puede detectar el error, o bien puede que el otro lo encuentre.

CADP – CORRECCION DE PROGRAMAS VERIFICACION

Es el proceso de controlar que se cumplan las pre y post condiciones del mismo.

Para determinar la corrección de un programa puedo utilizar una o varias técnicas de corrección la cantidad de veces necesarias hasta que le programa sea correcto

- Testing
- Debugging
- Walkthroughs
- Verificación



Conceptos de Algoritmos Datos y Programas

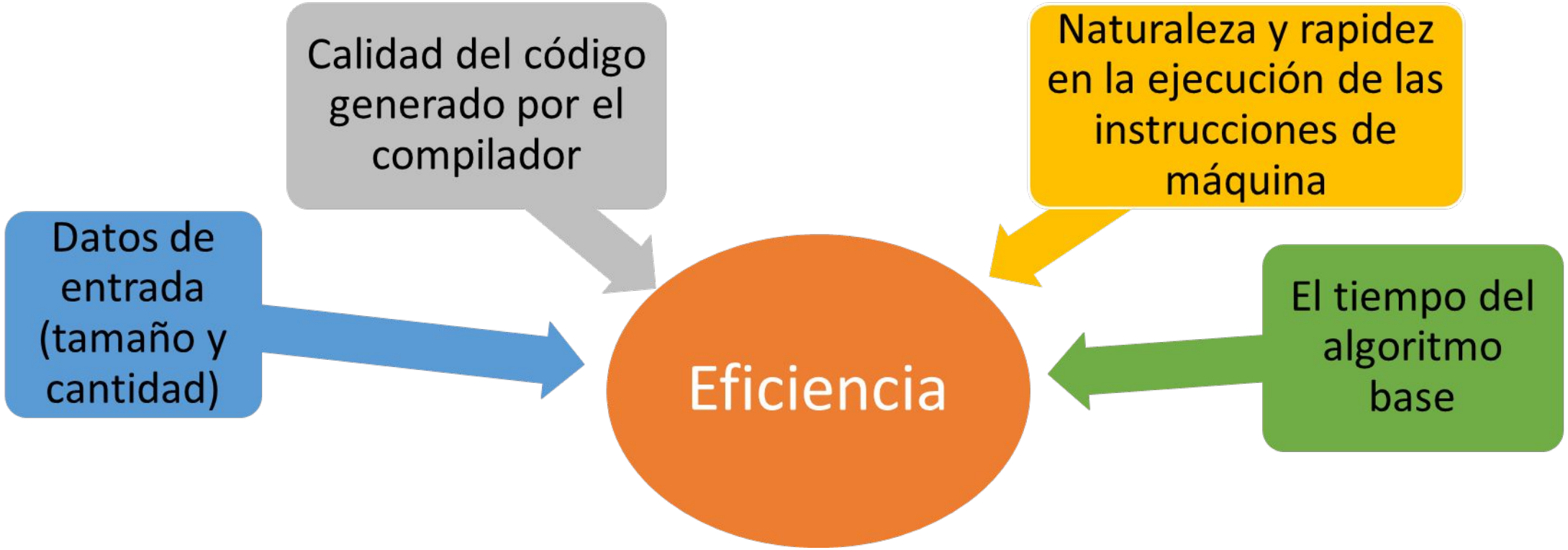
CADP – **TEMAS**

- Eficiencia de Programas

CADP – EFICIENCIA DE PROGRAMAS

Una vez que se obtiene un algoritmo y se verifica que es correcto, es importante determinar la eficiencia del mismo.

El análisis de la eficiencia de un algoritmo estudia el tiempo de ejecución de un algoritmo y la memoria que requiere para su ejecución.



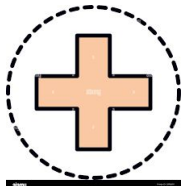
CADP – EFICIENCIA DE PROGRAMAS

El análisis de la eficiencia de un algoritmo estudia el tiempo de ejecución de un algoritmo y la memoria que requiere para su ejecución.

Los factores que afectan la eficiencia de un programa

Como se miden?

MEMORIA: Se calcula (como hemos visto previamente) teniendo en cuenta la cantidad de bytes que ocupa la declaración en el programa de:



- constante/s
- variable/s global/es
- variable/s local al programa/es

TIEMPO DE EJECUCION: puede calcularse haciendo un análisis empírico o un análisis teórico del programa.

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

El tiempo de un algoritmo puede definirse como una función de entrada:

- Existen algoritmos que el tiempo de ejecución tiempo de ejecución no depende de las características de los datos de entrada sino de la cantidad de datos de entrada o su tamaño.

- Existen otros algoritmos el tiempo de ejecución es una función de la entrada “específica”, en estos casos se habla del tiempo de ejecución del “peor” caso. En estos casos, se obtiene una cota superior del tiempo de ejecución para cualquier entrada

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

Para medir el tiempo de ejecución se puede realizar un análisis empírico o un análisis teórico

ANALISIS EMPIRICO

Requiere la implementación del programa, luego ejecutar el programa en la máquina y medir el tiempo consumido para su ejecución.

Fácil de realizar.

Obtiene valores exactos para una máquina determinada y unos datos determinados.

Completamente dependiente de la máquina donde se ejecuta

Requiere implementar el algoritmo y ejecutarlo repetidas veces (para luego calcular un promedio).

CADP – EFICIENCIA DE PROGRAMAS

T. de EJECUCION

Para medir el tiempo de ejecución se puede realizar un análisis empírico o un análisis teórico

ANALISIS TEORICO

Implica encontrar una cota máxima (“peor caso”) para expresar el tiempo de nuestro algoritmo, sin necesidad de ejecutarlo.

A partir de un programa correcto, se obtiene el tiempo teórico del algoritmo y luego el orden de ejecución del mismo. Lo que se compara entre algortimos es el orden de ejecución.

T(n) = 4 **UT**	O(n) = **C (constante)**
T(n) = (20 + N) **UT**	O(n) = **N**
T(n) = (20 + log N) **UT**	O(n) = **log N**
T(n) = (N * N) = N^2 **UT**	O(n) = **N^2**

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

Para medir el tiempo de ejecución se puede realizar un análisis empírico o un análisis teórico

ANALISIS TEORICO

Dado un algoritmo que es correcto se calcula el tiempo de ejecución de cada una de sus instrucciones. **Para eso se va a considerar**:

Sólo las **instrucciones elementales** del algoritmo: asignación, y operaciones aritmético/lógicas.

Una instrucción elemental utiliza un tiempo constante para su ejecución, independientemente del tipo de dato con el que trabaje. **1UT.**

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

```
Program uno;

var
 aux,temp,x: integer;

Begin
  (1) aux:= 58;
  (2) aux:= aux * 5;
  (3) temp:= aux;
  (4) read (x);
End.
```

Qué ocurre cuando hay estructuras de control?

El tiempo de ejecución de un algoritmo que NO tiene estructuras de control está dado por:

T (alg) = T(1) + T(2) + T(3) + T(4)

T(1)= asignación = 1UT

T(2)= multiplicación + asignación = 2UT

T(3)= asignación = 1UT

T(4)= no se considera

T (alg) = 4UT

CADP – EFICIENCIA DE PROGRAMAS

T. de EJECUCION

El tiempo de ejecución del **IF**

Tiempo de evaluar la condición + tiempo del cuerpo.

Si hay else, Tiempo de evaluar la condición + max(then,else).

El tiempo de ejecución del **FOR**

(3N + 2) + N(cuerpo del for).
N representa la cantidad de veces que se ejecuta el for.

El tiempo de ejecución del **WHILE**

C(N+1) + N(cuerpo del while).
N representa la cantidad de veces que se ejecuta el while.(N>=0)
C cantidad de tiempo en evaluar la condición

El tiempo de ejecución del **REPEAT UNTIL**

C(N) + N(cuerpo del repeat).
N representa la cantidad de veces que se ejecuta el repeat. (N>0)
C cantidad de tiempo en evaluar la condición

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3) + T(4)

```
Program uno;
var
 aux,temp,x: integer;

Begin
 (1) aux:= 58;
 (2) aux:= aux * 5;
 (3) if (aux > 45) and (aux <300)) then
       begin
        temp:= aux – 5;
        x:= temp + aux + 2;
       end;
 (4) x:= x * 10;
end;
```

T(1)= asignación = 1UT

T(2)= multiplicación + asignación = 2UT

T(3)= IF= evaluar la condición + cuerpo=
= (1) + (1) + (conector) = 3 +
resta + asignación
suma + suma + asignación
= 3UT + 2UT + 3UT = 8UT

T(4)= multiplicación + asignación = 2UT

T (alg) = 13 UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2)

```
Program uno;
var
 aux,temp,x: integer;

Begin
 (1) read(aux);
 (2) if (aux > 45) then
       begin
        temp:= aux - 5;
        x:= temp;
       end
     else
      aux:= aux + 1 * (aux MOD 2);
end;
```

T(1)= el read no se tiene en cuenta.

T(2)= IF= evaluar la condición + max(then,else)

cond= 1 +
then= 2 + 1 = 3
else= 4

= 1UT + max(3,4) = 5UT

T (alg) = 5 UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3)

```
Program uno;
var
 i,temp,x: integer;

Begin
 (1) aux:= 8;
 (2) for i:= 1 to 5 do
       begin
        x:= aux;
        aux:= aux + 5;
       end

 (3) aux:= aux + 1;
end;
```

T(1)= asignación 1 UT.

T(2)= **for=** 3N+2 + N(cuerpo)

N= 5
(3*5) +2 + 5(cuerpo)
17 + 5(cuerpo)
cuerpo = 1 + 2 = 3UT

>> 17 + 5(3) =

= 17 + 15 = 32UT

T(3)= 1 + 1 = 2 UT.

T (alg) = 35 UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3)

```
Program uno;
var
 i,temp,x: integer;

Begin
 (1) aux:= 8;
 (2) for i:= 4 to 9 do
       begin
        x:= aux;
        aux:= aux + 5;
       end

 (3) aux:= aux + 1;
end;
```

T(1)= asignación 1 UT.

T(2)= **for=** 3N+2 + N(cuerpo)

N= 6 (LS-LI + 1)
(3*6) +2 + 6(cuerpo)
20 + 6(cuerpo)
cuerpo = 1 + 2 = 3UT

>> 20 + 6(3) =

= 20 + 18 = 38UT

T(3)= 1 + 1 = 2 UT.

T (alg) = 41 UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3)

```
Program uno;
var
 i,temp,x: integer;

Begin
 (1) aux:= 0;
 (2) while (aux < 5) do
       begin
        x:= aux;
        aux:= aux + 1;
       end

 (3)  aux:= aux + 1;
end;
```

T(1)= asignación 1 UT.

T(2)= while= C(N+1)+ N(cuerpo)

C= 1
N = 5
1*(6) + 5(cuerpo)
cuerpo = 1 + 2 = 3UT

>> 6 + 5(3) =

= 6 + 15 = 21UT

T(3)= 1 + 1 = 2 UT.

T (alg) = 24 UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3)

```
Program uno;
var
 i,temp,x: integer;

Begin
 (1) read(aux);
 (2) while (aux < 5) do
       begin
        x:= aux;
        aux:= aux + 1;
       end

 (3)  aux:= aux + 1;
end;
```

T(1)= read no se cuenta 0 UT.

T(2)= while= C(N+1)+ N(cuerpo)

C= 1
N = ???
1*(N+1) + N(cuerpo)
cuerpo = 1 + 2 = 3UT

>> N+1 + N(3) =

= N + 1 + 3N = 4N + 1 UT

T(3)= 1 + 1 = 2 UT.

T (alg) = 0 + 4N +1 + 2 = (4N + 3) UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3)

```
Program uno;
var
 i,temp,x: integer;

Begin
 (1) aux:=0;
 (2) while (aux >= 0) and (aux<5) do
       begin
        x:= aux;
        aux:= aux + 1;
       end

 (3)  aux:= aux + 1;
end;
```

T(1)= 1 UT.

T(2)= while= C(N+1)+ N(cuerpo)

C= (1) + (1) + conector = 3
N = 5
3*(5+1) + 5(cuerpo)
cuerpo = 1 + 2 = 3UT

>> 18 + 5(3) =

= N + 18 + 15 = 33 UT

T(3)= 1 + 1 = 2 UT.

T (alg) = 1 + 33 + 2 = 36 UT

CADP – EFICIENCIA DE PROGRAMAS T. de EJECUCION

ANALISIS TEORICO

T (alg) = T(1) + T(2) + T(3)

```
Program uno;
var
 i,temp,x: integer;

Begin
 (1) aux:=0;
 (2) repeat
        x:= aux;
        aux:= aux + 1;
     until (aux > 5)

 (3) aux:= aux + 1;
end;
```

T(1)= 1 UT.

T(2)= repeat= C(N)+ N(cuerpo)

C= 1
N = 5
1*(5) + 5(cuerpo)
cuerpo = 1 + 2 = 3UT

>> 5 + 5(3) =

= N + 5 + 15 = 20 UT

T(3)= 1 + 1 = 2 UT.

T (alg) = 1 + 20 + 2 = 23 UT