

Detecció d'errors en programmes

PRO1

Josep Carmona, Lluís Padró



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Index

- Errors de compilació
- Errors d'execució
 - Resultats incorrectes
 - Execucions que avorten
- Exemple 1: Depuració d'un programa pas a pas
- Exercici individual

Introducció

- Els errors en un programa poden tenir diferents causes, i cal usar diferents estratègies per resoldre'ls.
- Hi ha **dos tipus** d'errors
 - Errors de compilació
 - Errors d'execució

Errors de compilació

- Ens els dóna el compilador
- Són deguts a algun error sintàctic o semàntic en el programa
- Causen que la compilació no generi cap executable.

ESTRATÈGIA DE RESOLUCIÓ:

- **Llegir bé l'error** del compilador (el primer que dóna), veure en quina línia es troba, i arreglar el codi en consonància.

Errors d'execució (i)

- El compilador genera l'executable sense errors, però a l'executar el programa els **resultats** no són els **esperats**.
- Poden ser **sistemàtics** (passen sempre) o **ocasionals** (passen només per certes dades d'entrada)
- Normalment es deuen a que el programa no tracta algú cas (o ho fa incorrectament).
- Poden causar comportaments estranys (des d'un resultat incorrecte per pocs decimals fins a un avortament de l'execució)

5

Errors d'execució (ii)

ESTRATÈGIA DE RESOLUCIÓ (1):

6

- Si el programa produeix un **resultat incorrecte**:
- **Identificar** en quins casos el programa produeix el resultat incorrecte.
- **Analitzar** el codi per veure si el tractament d'aquests casos està ben dissenyat.
- Pot ser útil afegir **traces** al codi per seguir-ne l'evolució.

Errors d'execució (iii)

ESTRATÈGIA DE RESOLUCIÓ (2):

- Si l'execució del programa **avorta**:
- **Identificar** en quins casos es produeix el problema.
- Pot ser útil afegir **traces** al codi per detectar **en quin punt** exacte té lloc l'avortament de l'execució.
- Un cop identificat el punt problemàtic, normalment ja es detecta el problema. Si no és així, es poden afegir traces per seguir l'evolució del programa en els moments anteriors a l'avortament

Exemple 1

Enunciat:

Escriure un programa que donat un natural $N > 0$ i una seqüència de N enters, digui si la seqüència es pot tallar en algun punt tal que les dues parts sumin el mateix.

Exemple:

La seqüència de 6 enters 4 3 2 2 2 1

Es pot tallar per la posició 2 en dues parts que sumen 7:

$$4+3 = 2+2+2+1$$


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

Primera versió del programa

```
int main() {
    int n;
    cin >> n;
    // llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v;
    v=0;
    int s=0
    for (i=0; i<n; ++i) {
        cin >> v[i];
    }

    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
    int se=0;
    int i=0;
    while (i<n and se!=s) {
        se = se+v[i];
        s = s-v[i];
    }

    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
}
```

Resultat de la compilació

```
$ p1++ -o exemple1.x exemple1.cc
```

```
exemple1.cc: In function 'int main()':
```

```
exemple1.cc:11:3: error: expected ';' before 'vector'
```

```
vector<int> v;  
^
```

```
exemple1.cc:12:3: error: 'v' was not declared in this scope
```

```
v=0;  
^
```

```
exemple1.cc:14:3: error: expected ',' or ';' before 'for'
```

```
for (i=0; i<n; ++i) {  
^
```

```
exemple1.cc:14:13: error: 'i' was not declared in this scope
```

```
for (i=0; i<n; ++i) {  
^
```

```
exemple1.cc:14:21: error: expected ';' before ')' token
```

```
for (i=0; i<n; ++i) {  
^
```

El compilador ens diu **on** ha trobat el problema (al main, línia 11, caràcter 3)

El problema és que ha trobat el tipus **vector** quan esperava un punt i coma.

El punt i coma falta a la línia 9, però el compilador no ho pot detectar fins que troba la següent instrucció i veu que no és el que esperava.

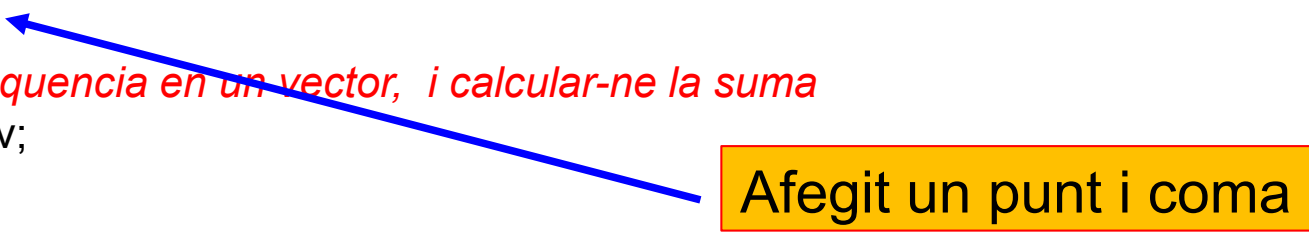
```
#include <iostream>
using namespace std;
```

Versió 2, corregint l'error

```
int main() {
    int n;
    cin >> n;
    // llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v;
    v=0;
    int s=0
    for (i=0; i<n; ++i) {
        cin >> v[i];
    }

    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
    int se=0;
    int i=0;
    while (i<n and se!=s) {
        se = se+v[i];
        s = s-v[i];
    }

    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
}
```



Afegit un punt i coma

Resultat de la compilació

```
$ g++ -o exemple1.x exemple1.cc
```

```
exemple1.cc: In function 'int main()':
```

```
exemple1.cc:11:3: error: 'vector' was not declared in this scope
```

```
vector<int> v;  
^
```

```
exemple1.cc:11:10: error: expected primary-expression before 'int'
```

```
vector<int> v;  
^
```

```
exemple1.cc:11:10: error: expected ';' before 'int'
```

```
exemple1.cc:12:3: error: 'v' was not declared in this scope
```

```
v=0;  
^
```

```
exemple1.cc:14:3: error: expected ',' or ';' before 'for'
```

```
for (i=0; i<n; ++i) {  
^
```

```
exemple1.cc:14:13: error: 'i' was not declared in this scope
```

```
for (i=0; i<n; ++i) {  
^
```

Sembla que encara hi ha un problema al mateix lloc.

Però el problema ara ja no és el punt i coma, sinó que no **vector** no està declarat.
Hem oblidat fer l'include de la llibreria on es defineix el tipus **vector**.

Versió 3, afegint l'include

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

```
int main() {
```

```
    int n;
```

```
    cin >> n;
```

```
    // llegir la seqüència en un vector, i calcular-ne la suma
```

```
    vector<int> v;
```

```
    v=0;
```

```
    int s=0
```

```
    for (i=0; i<n; ++i) {
```

```
        cin >> v[i];
```

```
    }
```

```
    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
```

```
    int se=0;
```

```
    int i=0;
```

```
    while (i<n and se!=s) {
```

```
        se = se+v[i];
```

```
        s = s-v[i];
```

```
    }
```

```
    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
```

```
        << " a la posició " << i << endl;
```

```
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
```

```
}
```

Afegit #include <vector>

Resultat de la compilació

```
$ g++ -o exemple1.x exemple1.cc
```

```
exemple1.cc: In function 'int main()':
```

```
exemple1.cc:13:4: error: no match for 'operator=' (operand types are 'std::vector<int>' and 'int')
```

```
  v=0;  
  ^
```

```
exemple1.cc:13:4: note: candidate is:
```

```
In file included from /usr/include/c++/4.8/vector:69:0,  
    from exemple1-3.cc:2:
```

```
/usr/include/c++/4.8/bits/vector.tcc:160:5: note: std::vector<_Tp, _Alloc>& std::vector<_Tp, _Alloc>::operator=(const  
std::vector<_Tp, _Alloc>&) [with _Tp = int; _Alloc = std::allocator<int>]
```

```
  vector<_Tp, _Alloc>:  
  ^
```

```
/usr/include/c++/4.8/bits/vector.tcc:160:5: note: no known conversion for argument 1 from 'int' to 'const  
std::vector<int>&'
```

```
exemple1.cc:15:3: error: expected ',' or ';' before 'for'
```

```
  for (i=0; i<n; ++i) {  
  ^
```

Ara hi ha un altre
error a la línia 13

L'error és que no està definida l'operació
d'assignació d'un enter a un vector
(instrucció v=0).
Aquesta instrucció no té sentit. Cal
eliminar-la.

14

Versió 4, eliminant v=0

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

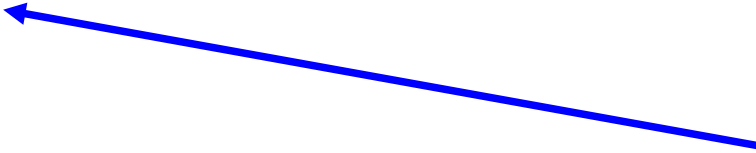
```
int main() {
    int n;
    cin >> n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v;
    int s=0
    for (i=0; i<n; ++i) {
        cin >> v[i];
    }
```

```
    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
```

```
    int se=0;
    int i=0;
    while (i<n and se!=s) {
        se = se+v[i];
        s = s-v[i];
    }
```

```
    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
}
```

Eliminada instrucció
errònia i innecessària
v=0



Resultat de la compilació

```
$ p1++ -o exemple1.x exemple1.cc
```

```
exemple1.cc: In function 'int main()':
```

```
exemple1.cc:14:3: error: expected ',' or ';' before 'for'
```

```
  for (i=0; i<n; ++i) {  
    ^
```

```
exemple1.cc:14:13: error: 'i' was not declared in this scope
```

```
  for (i=0; i<n; ++i) {  
    ^
```

```
exemple1.cc:14:21: error: expected ';' before ')' token
```

```
  for (i=0; i<n; ++i) {
```

Ara hi ha un altre error a la línia 14

Un altre cop, falta un punt i coma. En aquest cas ho detecta al trobar el «for» de la línia 14. L'error està a la instrucció anterior.

Versió 5, afegint el punt i coma

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

```
int main() {
```

```
    int n;
```

```
    cin >> n;
```

```
    // Llegir la seqüència en un vector, i calcular-ne la suma
```

```
    vector<int> v;
```

```
    int s=0;
```

```
    for (i=0; i<n; ++i) {
```

```
        cin >> v[i];
```

```
    }
```

```
    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
```

```
    int se=0;
```

```
    int i=0;
```

```
    while (i<n and se!=s) {
```

```
        se = se+v[i];
```

```
        s = s-v[i];
```

```
    }
```

```
    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
```

```
        << " a la posició " << i << endl;
```

```
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
```

```
}
```

Afegit un punt i coma

Resultat de la compilació

```
$ p1++ -o exemple1.x exemple1.cc
```

```
exemple1.cc: In function 'int main()':
```

```
exemple1.cc:14:8: error: 'i' was not declared in this scope
```

```
  for (i=0; i<n; ++i) {  
      ^
```

Ara hi ha un altre error a la línia 14, però més endavant (posició 8, on posa i=0)

En aquest cas, l'error està en que no hem especificat el tipus de la variable del bucle.

18

Versió 6, declarant la variable del for

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

```
int main() {
    int n;
    cin >> n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v;
    int s=0;
    for (int i=0; i<n; ++i) {
        cin >> v[i];
    }

    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
    int se=0;
    int i=0;
    while (i<n and se!=s) {
        se = se+v[i];
        s = s-v[i];
    }

    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
}
```

Afegit tipus de la variable del for

Resultat de la compilació

```
$ g++ -o exemple1.x exemple1.cc  
$ ls  
exemple1.cc  
exemple1.x
```

La compilació acaba sense cap error

S'ha creat l'executable exemple1.x

Exemple 1 (continuació)

- Un cop tenim l'executable, podem provar si el programa fa el que preteniem.
- Per fer-ho, cal executar-ho amb alguns jocs de proves i veure si el comportament es l'esperat.
- Començarem provant amb la seqüència de 6 elements: 4 3 2 2 2 1
- Com que $4+3 = 2+2+2+1$, el resultat esperat és que el programa ens digui que es pot dividir la seqüència en dues parts de suma set tallant per la posició 2.

Primera execució

```
$ ./exemple1.x exemple1.cc
```

```
6
```

```
4 3 2 2 2 1
```

Violació de segment (bolcat de la imatge del nucli)

- El programa no dona cap sortida, i ens trobem amb una execució avortada per violació de segment.
- Posarem punts de control al programa per localitzar el problema.

Versió 6, afegint punts de control

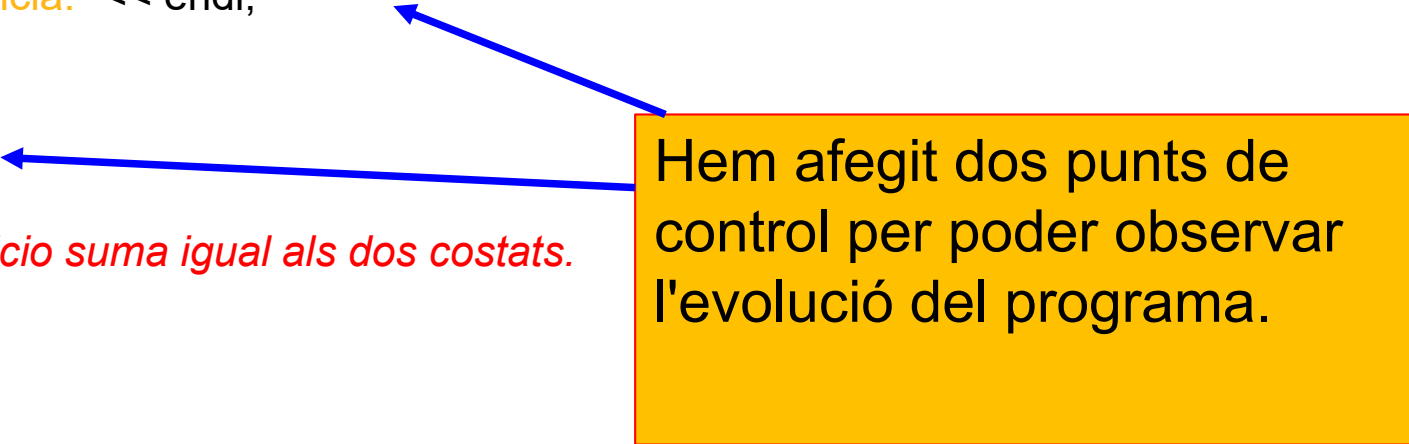
```
#include <iostream>
#include <vector>
using namespace std;
```

```
int main() {
    int n;
    cin>>n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v;
    int s=0;
    for (int i=0; i<n; ++i) {
        cout << "Llegint element " << i << " de la seqüència." << endl;
        cin >> v[i];
    }
    cout << "Seqüència llegida" << endl;

    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
    int se=0;
    int i=0;
    while (i<n and se!=s) {
        se = se+v[i];
        s = s-v[i];
    }

    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
```

23



Hem afegit dos punts de control per poder observar l'evolució del programa.

Execució amb els punts de control

```
$ g++ -o exemple1.x exemple1.cc
```

```
$ ./exemple1.x
```

```
6
```

```
4 3 2 2 2 1
```

Llegint element 0 de la seqüència.

Violació de segment (bolcat de la imatge del nucli)

- Veiem que l'error passa just a la primera volta del bucle (ja que el missatge de control només surt un cop)
- El problema està en que el vector `v` no té elements (ja que no n'hem declarat la mida) i per tant, intentar omplir-lo se surt de la memòria reservada per a la variable `v`

Versió 7, reservant espai pel vector

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

```
int main() {
    int n;
    cin>>n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v(n);
    int s=0;
    for (int i=0; i<n; ++i) {
        cout << "Llegint element " << i << " de la seqüència." << endl;
        cin >> v[i];
    }
    cout << "Seqüència llegida" << endl;
```

Declarem el vector de la mida necessària.

// per cada element, mirar si el tall en aquella posició suma igual als dos costats.

```
int se=0;
int i=0;
while (i<n and se!=s) {
    se = se+v[i];
    s = s-v[i];
}
```

```
if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
    << " a la posició " << i << endl;
else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
```

Execució del programa corregit

```
$ g++ -o exemple1.x exemple1.cc
```

```
$ ./exemple1.x exemple1.cc
```

```
6
```

```
4 3 2 2 2 1
```

Llegint element 0 de la seqüència.

Llegint element 1 de la seqüència.

Llegint element 2 de la seqüència.

Llegint element 3 de la seqüència.

Llegint element 4 de la seqüència.

Llegint element 5 de la seqüència.

Seqüència llegida

La seqüència es pot tallar en dues parts de suma 0 a la posició 0

La seqüència es llegeix correctament i el programa ja no avorta.

Però el resultat no és el que esperàvem (dues parts de suma 7 a la posició 2).
Afegirem més punts de control per veure què està passant.

Versió 7, amb punts de control addicionals

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

```
int main() {
    int n;
    cin >> n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v(n);
    int s=0;
    for (int i=0; i<n; ++i) {
        cout << "Llegint element " << i << " de la seqüència." << endl;
        cin >> v[i];
    }
    cout << "Seqüència llegida n=" << n << " s=" << s << endl;
```

// per cada element, mirar si el tall en aquella posició suma igual als dos costats.

```
int se=0;
int i=0;
while (i<n and se!=s) {
    se = se+v[i];
    s = s-v[i];
}
```

```
if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
    << " a la posició " << i << endl;
else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
```

Hem modificat aquest punt de control perquè doni més detalls sobre els valors de les variables.

Execució del programa amb el nou punt de control

```
$ p1++ -o exemple1.x exemple1.cc
```

```
$ ./exemple1.x exemple1.cc
```

```
6
```

```
4 3 2 2 2 1
```

Llegint element 0 de la seqüència.

Llegint element 1 de la seqüència.

Llegint element 2 de la seqüència.

Llegint element 3 de la seqüència.

Llegint element 4 de la seqüència.

Llegint element 5 de la seqüència.

Seqüència llegida n=5 s=0

La seqüència es pot tallar en dues parts de suma 0 a la posició 0

A l'acabar de llegir el vector, la variable s hauria de ser la suma dels seus elements, tal com diu el comentari d'abans del bucle.

Però hem oblidat fer que el bucle calculi la suma. Cal afegir les instruccions necessàries

Versió 8, sumant els elements

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

```
int main() {
    int n;
    cin >> n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v(n);
    int s=0;
    for (int i=0; i<n; ++i) {
        cout << "Llegint element " << i << " de la seqüència." << endl;
        cin >> v[i];
        s += v[i];
    }
    cout << "Seqüència llegida n=" << n << " s=" << s << endl;
```

Faltava acumular la suma dels elements a mida que es van llegint.

```
// per cada element, mirar si el tall en aquella posició suma igual als dos costats.
```

```
int se=0;
int i=0;
while (i<n and se!=s) {
    se = se+v[i];
    s = s-v[i];
}
```

```
if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
    << " a la posició " << i << endl;
else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
```

Execució de la versió 8

```
$ g++ -o exemple1.x exemple1.cc
```

```
$ ./exemple1.x exemple1.cc
```

```
6
```

```
4 3 2 2 2 1
```

Llegint element 0 de la seqüència.

Llegint element 1 de la seqüència.

Llegint element 2 de la seqüència.

Llegint element 3 de la seqüència.

Llegint element 4 de la seqüència.

Llegint element 5 de la seqüència.

Seqüència llegida n=5 s=14

Ara la variable s ja té el valor correcte després de llegir els elements del vector.

Pero el programa no imprimeix res més i es queda penjat, probablement en un bucle infinit.

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

```
int main() {
    int n;
    cin >> n;
    // Llegir la seqüencia en un vector, i calcular-ne la suma
    vector<int> v(n);
    int s=0;
    for (int i=0; i<n; ++i) {
        cout << "Llegint element " << i << " de la seqüencia." << endl;
        cin >> v[i];
        s += v[i];
    }
    cout << "Seqüencia llegida n=" << n << " s=" << s << endl;

    // per cada element, mirar si el tall en aquella posicio suma igual als dos costats.
    int se=0;
    int i=0;
    while (i<=n and se!=s) {
        se = se+v[i];
        s = s-v[i];
        cout << "provant tall a la posicio i=" << i << " s=" << s << " se=" << se << endl;
    }

    if (i<n) cout << "La seqüencia es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüencia NO es pot tallar en dues parts d'igual suma " << endl;
```

Afegim més punts de control al segon bucle.

Afegim un punt de control per observar què passa dins del segon bucle.

Execució amb els nous controls

```
$ p1++ -o exemple1.x exemple1.cc
```

```
$ ./exemple1.x exemple1.cc
```

```
6
```

```
4 3 2 2 2 1
```

Llegint element 0 de la sequencia.

Llegint element 1 de la sequencia.

Llegint element 2 de la sequencia.

Llegint element 3 de la sequencia.

Llegint element 4 de la sequencia.

Llegint element 5 de la sequencia.

Sequencia llegida n=5 s=14

provant tall a la posicio i=0 s=-39458 se=39479

provant tall a la posicio i=0 s=-39459 se=39480

provant tall a la posicio i=0 s=-39460 se=39481

Veiem que el programa està en un bucle infinit, degut a que la variable *i* sempre val zero.

Hem oblidat incrementar la *i* dins del while.


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

Versió 9, incrementant la *i*

```
int main() {
    int n;
    cin>>n;
    // Llegir la seqüència en un vector, i calcular-ne la suma
    vector<int> v(n);
    int s=0;
    for (int i=0; i<n; ++i) {
        cout << "Llegint element " << i << " de la seqüència." << endl;
        cin >> v[i];
        s += v[i];
    }
    cout << "Seqüència llegida n=" << n << " s=" << s << endl;

    // per cada element, mirar si el tall en aquella posició suma igual als dos costats.
    int se=0;
    int i=0;
    while (i<=n and se!=s) {
        se = se+v[i];
        s = s-v[i];
        cout << "provant tall a la posició i=" << i << " s=" << s << " se=" << se << endl;
        ++i;
    }

    if (i<n) cout << "La seqüència es pot tallar en dues parts de suma " << s
        << " a la posició " << i << endl;
    else cout << "La seqüència NO es pot tallar en dues parts d'igual suma." << endl;
}
```

33

Incrementem la posició de tall perquè vagi variant.

Execució de la nova versió

```
$ p1++ -o exemple1.x exemple1.cc
```

```
$ ./exemple1.x exemple1.cc
```

```
6
```

```
4 3 2 2 2 1
```

Llegint element 0 de la seqüència.

Llegint element 1 de la seqüència.

Llegint element 2 de la seqüència.

Llegint element 3 de la seqüència.

Llegint element 4 de la seqüència.

Llegint element 5 de la seqüència.

Seqüència llegida n=5 s=14

provant tall a la posició i=0 s=10 se=4

provant tall a la posició i=1 s=7 se=7

La seqüència es pot tallar en dues parts de suma 7 a la posició 2

La posició ja va canviant, i la suma de les dues meitats es modifica en adequadament.

El resultat és el que esperàvem

Últims passos

- Sembla que hem corregit els errors que hi havia al programa, però podrien haver-n'hi més.
- Falta assegurar-se que el programa funciona correctament en altres casos, provant amb més jocs de proves, com p.e.:
 - seqüències que no es puguin tallar en dues parts que sumin igual
 - seqüències amb números negatius
 - seqüències en que la suma sigui zero
 - etc.
- Finalment, caldria treure tots els missatges de control, per tal que el programa doni només el resultat que es demana, i no tota l'evolució del programa.

Exercici

Useu el codi proporcionat zigazaga.cc per resoldre el problema *P71711 Zig-zag* de la llista de matrius, usant la metodologia de depuració presentada per corregir-ne els errors.