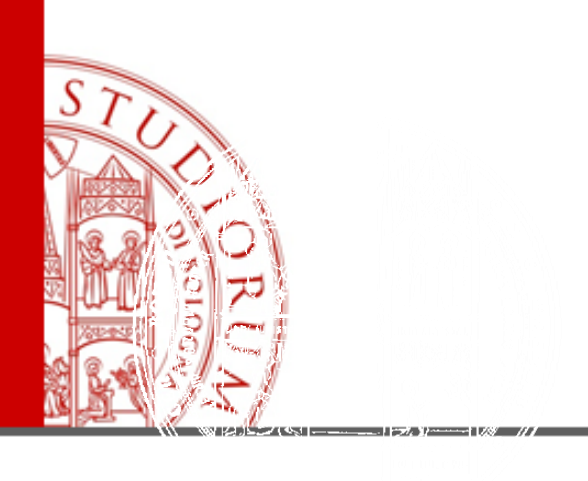


Università degli Studi di Bologna
Scuola di Ingegneria e Architettura

Capitolo 9 - LABORATORIO

Case study: interprete per "Small C"

(Estratto dal progetto d'esame di Marco Cova – 2003)

Corso di Laurea Magistrale in Ingegneria Informatica

Anno accademico 2020/2021

Prof. ENRICO DENTI

Dipartimento di Informatica – Scienza e Ingegneria (DISI)

LINGUAGGIO: OVERVIEW

Sottoinsieme del C

- Un solo tipo di dati: `int` (inclusi boolean)
- Operatori aritmetici, boolean, relazionali e assegnamento
- Istruzioni di selezione (`if`) e di iterazione (`while`)
- Blocchi innestati con (gerarchia di) environment
- Environment multipli
  - un environment globale + tanti environment locali ai blocchi
  - tempo di vita di un environment pari al corrispondente blocco
  - la ricerca di un simbolo avviene a partire dall'environment corrente
  - limite inferiore della catena: l'environment globale

LINGUAGGIO: GRAMMATICA (1)

CompilationUnit := {Declaration} {Statement}
Declaration := Type Identifier ['=' Expression] ';'
Type := 'int'
Identifier := IdentifierHead | IdentifierHead IdentifierTail
IdentifierHead := Letter | '_'
IdentifierTail := (IdentifierHead | Digit) {IdentifierTail}
Letter := 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'
NonZeroDigit := '1' | ... | '9'
ZeroDigit := '0'
Digit := ZeroDigit | NonZeroDigit
Number := NonzeroDigit {Digit} ['.' {Digit}]
| [ZeroDigit] '.' {Digit}

LINGUAGGIO: GRAMMATICA (2)

Statement := ';' | Expression ';' | BlockStatement
| IfStatement | WhileStatement

BlockStatement := '{' {Declaration} {Statement} '}'

IfStatement := 'if' '(' Expression ')' Statement ['else' Statement]

WhileStatement := 'while' '(' Expression ')' Statement

Expression := *espressioni aritmetiche e booleane (v. oltre)*

Expression := Identifier { '=' Expression }

Expression := '(' Expression ')'

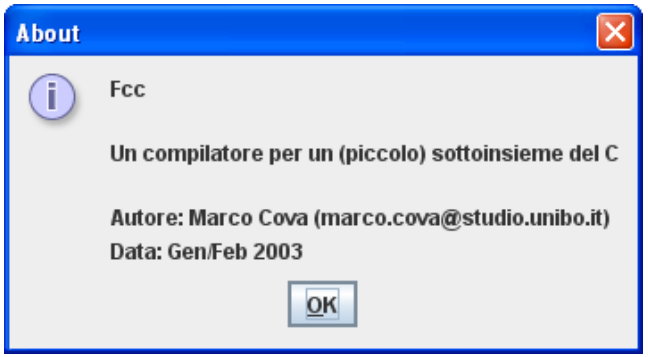
Expression := Number

LINGUAGGIO: GRAMMATICA (3)

Sottolinguaggio espressioni aritmetiche e booleane: *(non LL(1)):*

Expression := E1 | Expression '||' E1	*or logico*
E1 := E2 | E1 '&&' E2	*and logico*
E2 := E3 | E2 ('<' | '<=' | '>' | '>=' | '==' | '!=') E3	*relazionali*
E3 := E4| E3 ('+' | '–') E4	*additivi*
E4 := E5 | E4 ('*' | '/' | '%') E5	*moltiplicativi*
E5 := E6 | '!' E5	*negazione*
E6 := Number | Identifier | '(' Expression ')'	*fattori*

(per E6 si potrebbe accettare anche –Number*)*



LINGUAGGIO: GRAMMATICA (4)

Versione LL(1) in EBNF:

Expression := E1 { '||' E1 }	*or logico*
E1 := E2 { '&&' E2 }	*and logico*
E2 := E3 { ('<' | '<=' | '>' | '>=' | '==' | '!=') E3}	*relazionali*
E3 := E4 { ('+' | '-') E4 }	*additivi*
E4 := E5 { ('*' | '/' | '%') E5 }	*moltiplicativi*
E5 := E6 | '!' E5	*negazione*
E6 := Number | Identifier | '(' Expression ')'	*fattori*

(per E6 si potrebbe accettare anche -Number*)*

IMPLEMENTAZIONE (1)

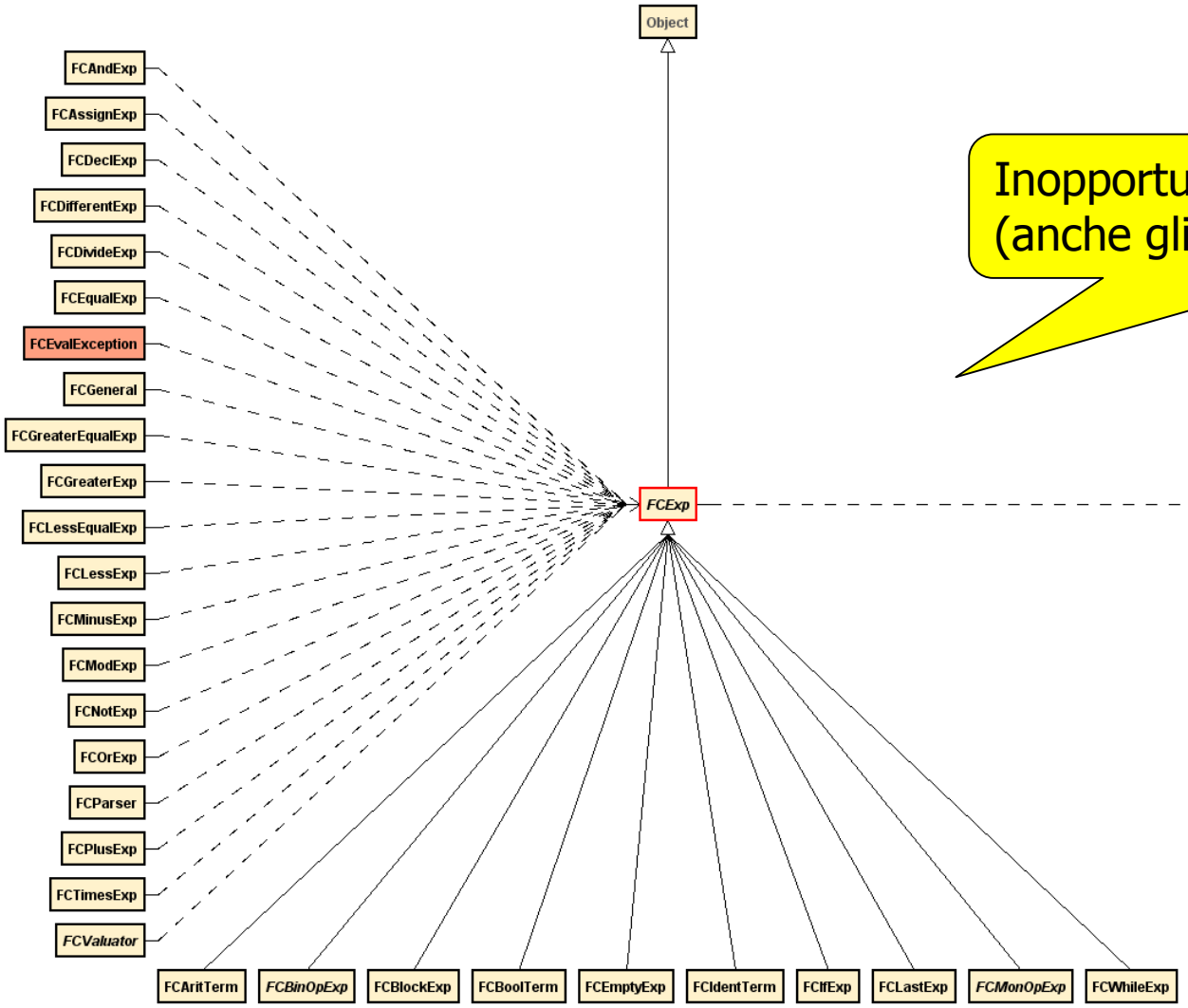
Analisi ricorsiva discendente

- parseCompilationUnit()
- parseDeclarations()
- parseStatement()
- parseBlock(), parseIf(), parseWhile()
- parseExp() *[ridenominata anche parseE0()],*
  parseE1(), parseE2(), parseE3(), parseE4(), parseE5(), parseE6()

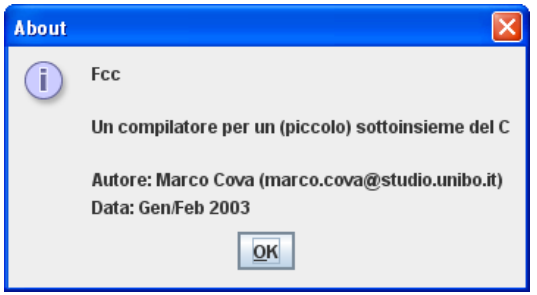
Tipi di token

- IDENT
- OPERATOR
- SLETTER
- CONSTANT
- SEPARATOR
- KEYWORD

IMPLEMENTAZIONE (2)

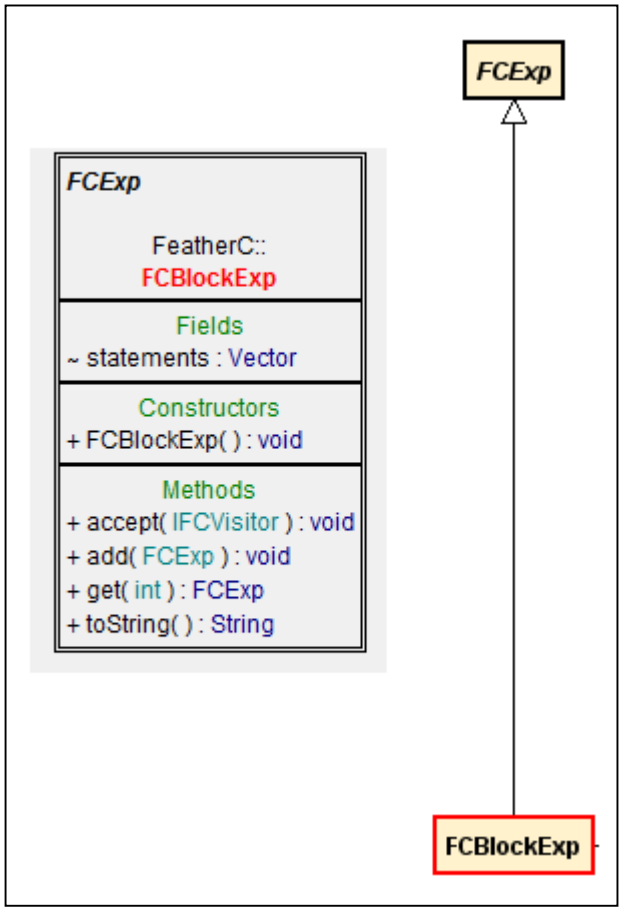


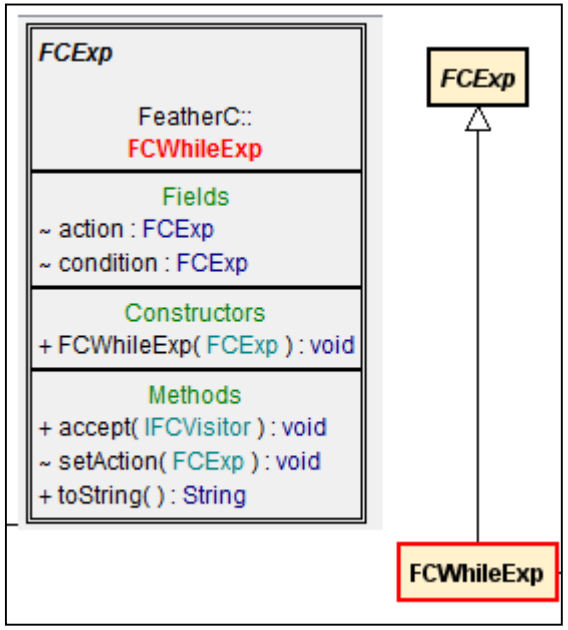
Inopportuno chiamare tutto FCExp..
(anche gli statement!)

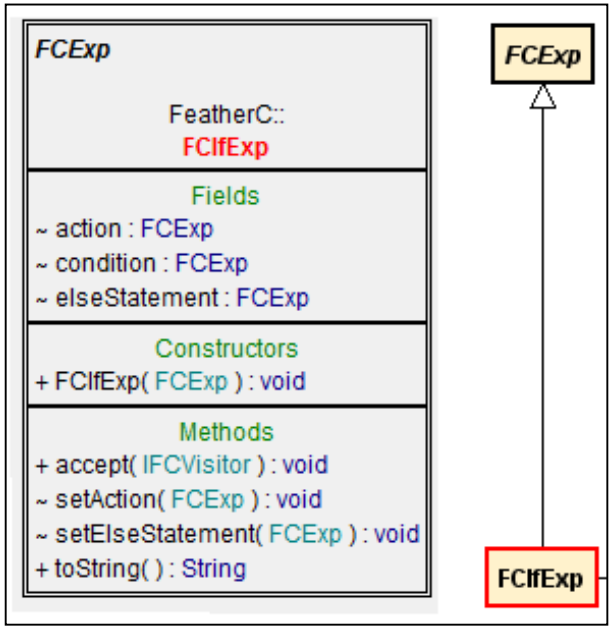


IMPLEMENTAZIONE (3)

Alcuni punti interessanti…

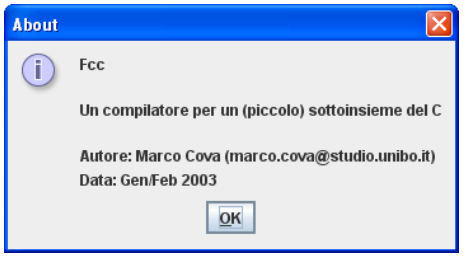






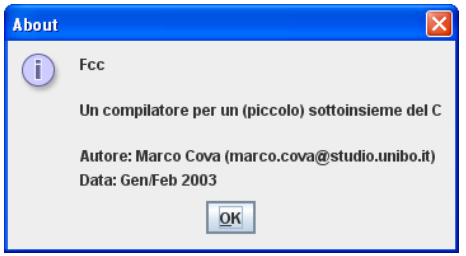
TASSONOMIA: ALCUNI ESTRATTI (1)

```
abstract class FCExp {  // espressioni e statement!
    abstract void accept(IFCVisitor v);
}
class FCLastExp extends FCExp {
    public String toString() {return "LAST_EXP";}
    public void accept(IFCVisitor v) {}
}
class FCEmptyExp extends FCExp {
    public String toString() { return "EMPTY_EXP";}
    public void accept(IFCVisitor v) {}
}
...
```



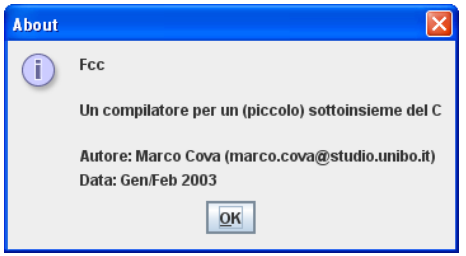
TASSONOMIA: ALCUNI ESTRATTI (2)

```
class FCBlockExp extends FCExp {
   Vector statements;
   public FCBlockExp() { statements = new Vector(); }
   public void accept(IFCVisitor v) {v.visit(this);}
   public String toString() { ... }
   public FCExp get(int index) {
      return (FCExp) statements.elementAt(index);
   }
   public void add(FCExp exp) { statements.add(exp); }
}
```



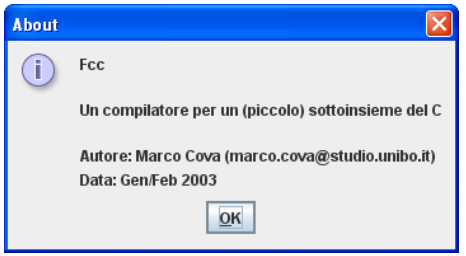
TASSONOMIA: ALCUNI ESTRATTI (3)

```
class FCIfExp extends FCExp {
   FCExp cond, action = null, elseStat=null;
   public FCIfExp(FCExp cond) { this.cond = cond; }
   public void accept(IFCVisitor v) {v.visit(this);}
   public String toString() { ... }
   void setAction(FCExp a) { this.action = a; }
   void setElseStatement(FCExp s) {this.elseStat=s; }
}
class FCWhileExp extends FCExp {
   FCExp cond, action = null;
   public FCWhileExp(FCExp cond) { this.cond = cond;}
   public void accept(IFCVisitor v) {v.visit(this);}
   public String toString() {...}
   void setAction(FCExp a) {
   this.action = a; }
}
```



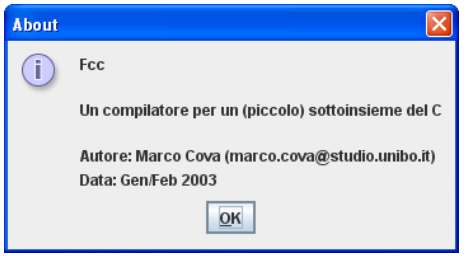
IL PARSER: ESTRATTO (1)

```
public class FCParser extends Thread {
 public void run() {
  FCExp exp = null; boolean end = false;
   try { currentToken = getNextToken(); }
   catch (FCException e) { end = true; }
   try {
   while(!(currentToken instanceof FCLastToken) && !stop) {
      exp = parseCompilationUnit();
      if (exp == null) {
            System.out.println("Parsing impossibile");
            continue;
      }
      if (exp instanceof FCEmptyExp) continue;
      bufferExp.put(exp);
   }
   ...
 }
```



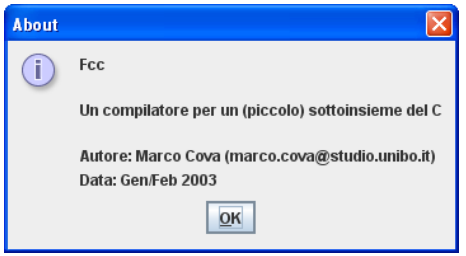
IL PARSER: ESTRATTO (2)

```
FCExp parseCompilationUnit() throws FCException {
   FCExp exp = null;
   exp = parseDeclarations();
   if (exp != null) return exp;
   exp = parseStatement();
   if (exp != null) return exp;
   return exp;
}
```



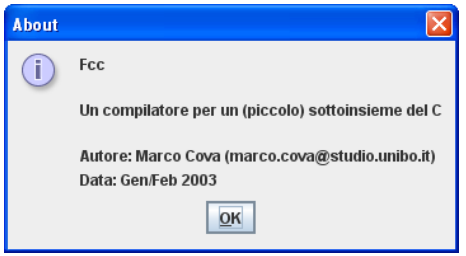
IL PARSER: ESTRATTO (3)

```
FCExp parseStatement() throws FCException {
   FCExp st = null;
   if (currentToken.type == FCTokenType.SEPARATOR &&
       currentToken.name.equals(";")) {
          st = new FCEmptyExp();
          currentToken = getNextToken();
   }
   else
   if (currentToken.type == FCTokenType.SEPARATOR &&
       currentToken.name.equals("{")) {
          currentToken = getNextToken();
          st = parseBlock();
   }
   else
   ...
```



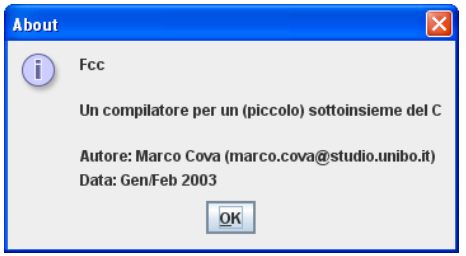
IL PARSER: ESTRATTO (4)

```
...
if (currentToken.type == FCTokenType.KEYWORD &&
    currentToken.name.equals("if")) {
        currentToken = getNextToken();
        st = parseIf();
}
else
if (currentToken.type == FCTokenType.KEYWORD &&
    currentToken.name.equals("while")) {
        currentToken = getNextToken();
        st = parseWhile();
}
else { // espressioni
...
```



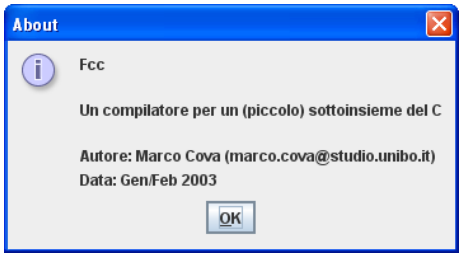
IL PARSER: ESTRATTO (5)

```
...
else { // espressioni
   st = parseExp();
   if (currentToken.type != FCTokenType.SEPARATOR ||
        !currentToken.name.equals(";")) {
            throw new FCParserException(
             "Missing expected ';'", currentToken);
     }
     currentToken = getNextToken();
}
endDeclarations = true;
return st;
}
```



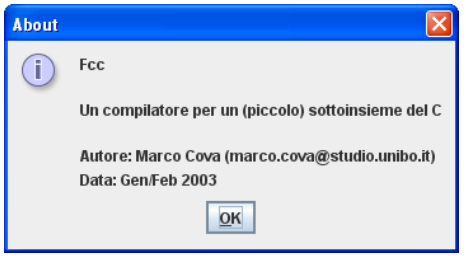
IL PARSER: ESTRATTO (6)

```
FCExp parseIf() throws FCException {
  FCIfExp ifExp;
  if (currentToken.type != FCTokenType.SEPARATOR ||
      !currentToken.name.equals("(")) {
      throw new FCParserException("Expected '(' ", …);
  }
  currentToken = getNextToken();
  FCExp cond = parseExp();
  if (cond == null) return null;
  if (currentToken.type != FCTokenType.SEPARATOR ||
      !currentToken.name.equals(")")) {
      throw new FCParserException("Expected ')' ", …);
  }
  currentToken = getNextToken();
  ...
```



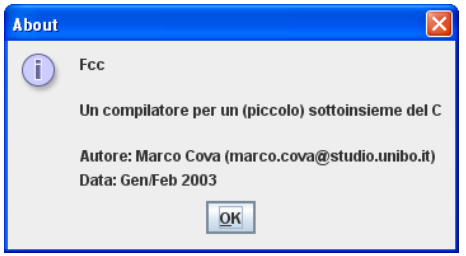
IL PARSER: ESTRATTO (7)

```
... // l'oggetto IfExp ha cond e action
ifExp = new FCIfExp(cond);
FCExp action = parseStatement();
if (action == null) {throw new
        FCParserException("Error after ')'",…);
}
ifExp.setAction(action);
if (currentToken.name.equals("else")) {
     currentToken = getNextToken();
     FCExp elseStat = parseElse();
     if (elseStat == null) { throw new
        FCParserException("Error after 'else'",…);
     }
     ifExp.setElseStatement(elseStat);
}
return ifExp;
}
```



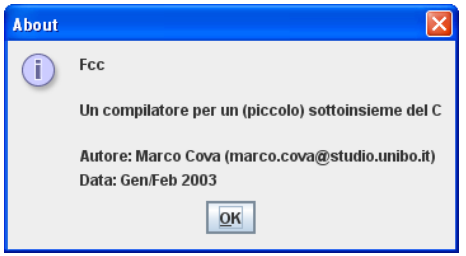
IL PARSER: ESTRATTO (8)

```
FCExp parseElse() throws FCException {
  FCIfExp elseExp = null;
  if (currentToken.name.equals("if")) {
       currentToken = getNextToken();
       elseExp = parseIf();     // if innestati
  }
  else
  elseExp = parseStatement(); // altro statement != if
  return elseExp;
}
```



IL PARSER: ESTRATTO (9)

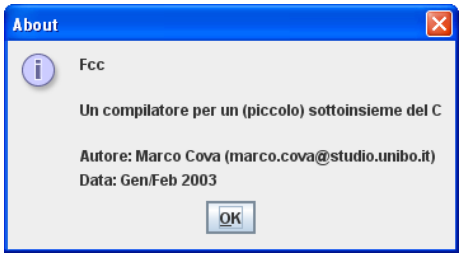
```
FCExp parseWhile() throws FCException {
  FCWhileExp whileExp;
  if (currentToken.type != FCTokenType.SEPARATOR ||
      !currentToken.name.equals("(")) { throw new
         FCParserException("Missing expected '('",…);
  }
  currentToken = getNextToken();
  FCExp cond = parseExp();
  if (cond == null) return null;
  if (currentToken.type != FCTokenType.SEPARATOR ||
      !currentToken.name.equals(")")) { throw new
    FCParserException("Missing expected ')'", …);
  }
  currentToken = getNextToken();
  ...
```



IL PARSER: ESTRATTO (10)

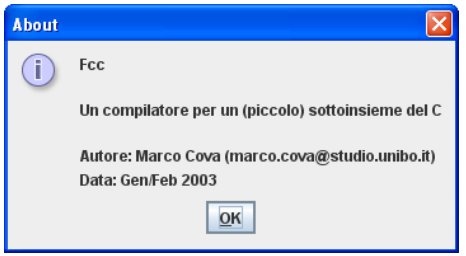
```
  ...
  whileExp = new FCWhileExp(cond);
  FCExp action = parseStatement();
  if (action == null) { throw new
     FCParserException("Parse error after ')'",...);
  }
  whileExp.setAction(action);
  return whileExp;
}
```

...analogamente per gli altri metodi..



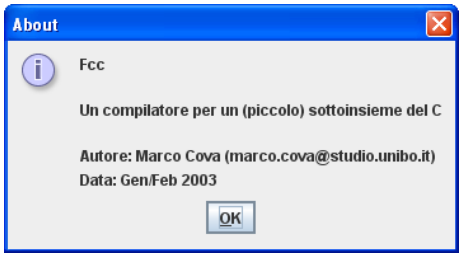
IL VALUTATORE: ESTRATTO (1)

```
public void visit(FCDeclExp e) {
  HashMap currentEnv;
  e.left.accept(this);    String lres = (String) getRes();
  e.right.accept(this);   Integer rres = (Integer) getRes();
  if (blockLevel == -1) // stato del valutatore (visitor)
      currentEnv = globalEnv;
  else
      currentEnv = (HashMap) localEnv.get(blockLevel);
  if (currentEnv.containsKey(lres)) {
      errorMsgs += "Already defined symbol '" + lres + "'";
      parser.stop = true; stop = true; errors++;
  }
  currentEnv.put(lres, new Ident(rres, e.type));
  res.add(rres);
}
```



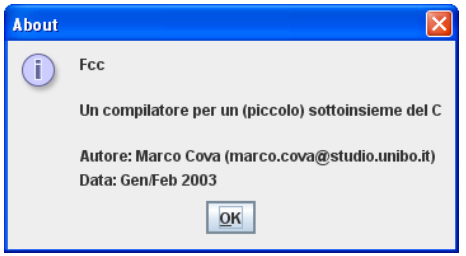
IL VALUTATORE: ESTRATTO (2)

```
public void visit(FCIdentVal e) {
  int i;
  Integer d = null;
  for (i = blockLevel; i >= -1; i--) {
    HashMap environment =
     (HashMap) ((i == -1) ? globalEnv : localEnv.get(i));
    Ident id = (Ident) environment.get(e.name);
    if (id != null)  d = (Integer) id.getValue();
    if (d != null) break;
  }
  if (i >= -1) res.add(new Integer(d.intValue()));
  else {
      parser.stop = true; stop = true; errors++;
  }
}
```



IL VALUTATORE: ESTRATTO (3)

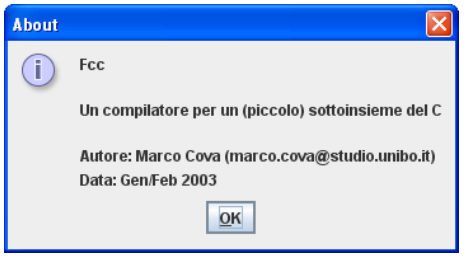
```
public void visit(FCWhileExp e) {
  e.condition.accept(this);
  int d = ((Integer) getRes()).intValue();
 // se la condizione è vera…
  while (d != 0) {
      e.action.accept(this); // eseguire l'azione
      e.condition.accept(this); // riverificare cond
      d = ((Integer)getRes()).intValue();
  }
}
```



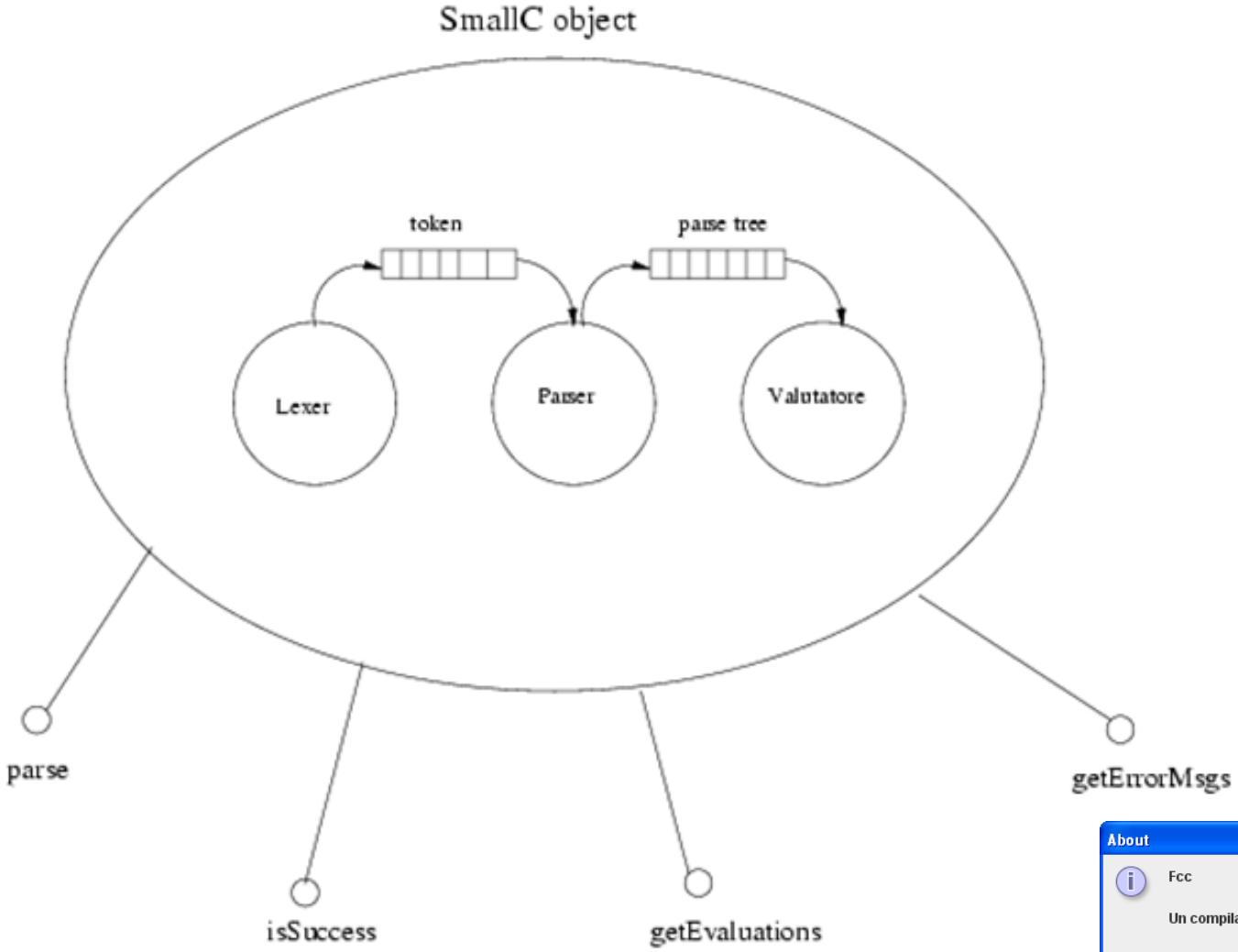
IL VALUTATORE: ESTRATTO (4)

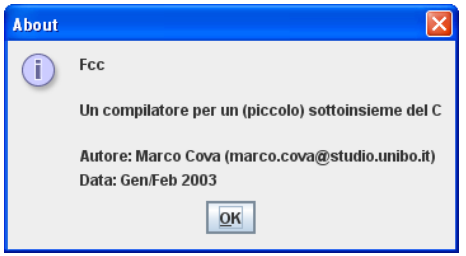
```
public void visit(FCIfExp e) {
  e.condition.accept(this);
  int d = ((Integer) getRes()).intValue();
 // se la condizione è vera…
  if (d != 0) {
      e.action.accept(this); // eseguire l'azione
  } // altrimenti…
  else if (e.elseStatement != null) {
    // eseguire il ramo else (se presente)
    e.elseStatement.accept(this);
  }
}
```

bisogna creare ecosistemi che dialogano senza esplodere



ARCHITETTURA

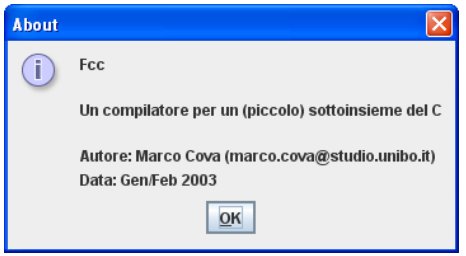




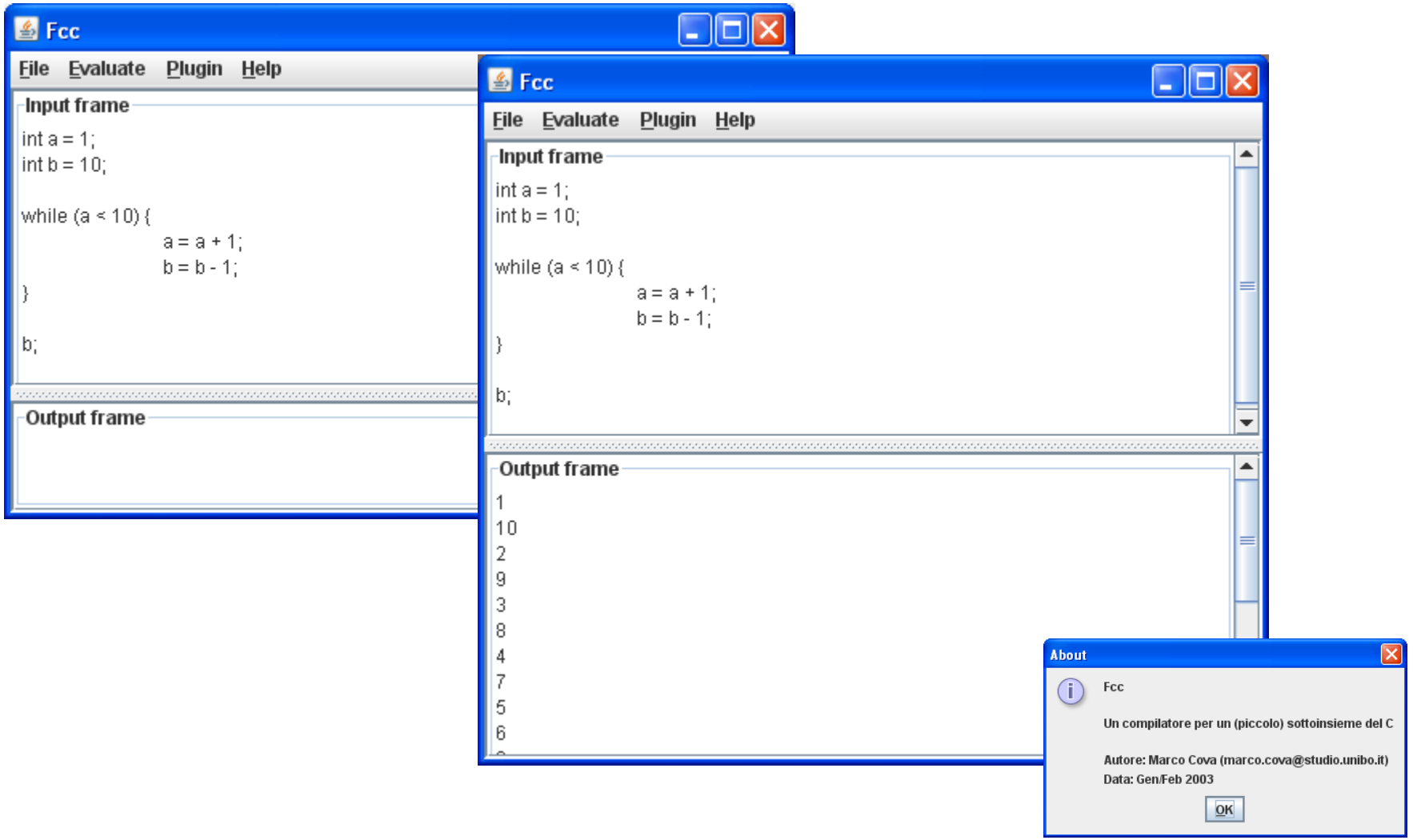
USO EMBEDDED da Java

```
SmallC engine = new SmallC();
SolveInfo info = engine.solve(
   new BufferedReader(new FileReader("source.sc")));
if (info.isSuccess()) {
     Vector sol = solInfo.getEvaluations();
     for (int j = 0; j < sol.size(); j++)
          System.out.println(sol.elementAt(j));
} else {
     System.out.println("***There were errors:\n" +
                              solInfo.getErrorMsgs()    );
}
```

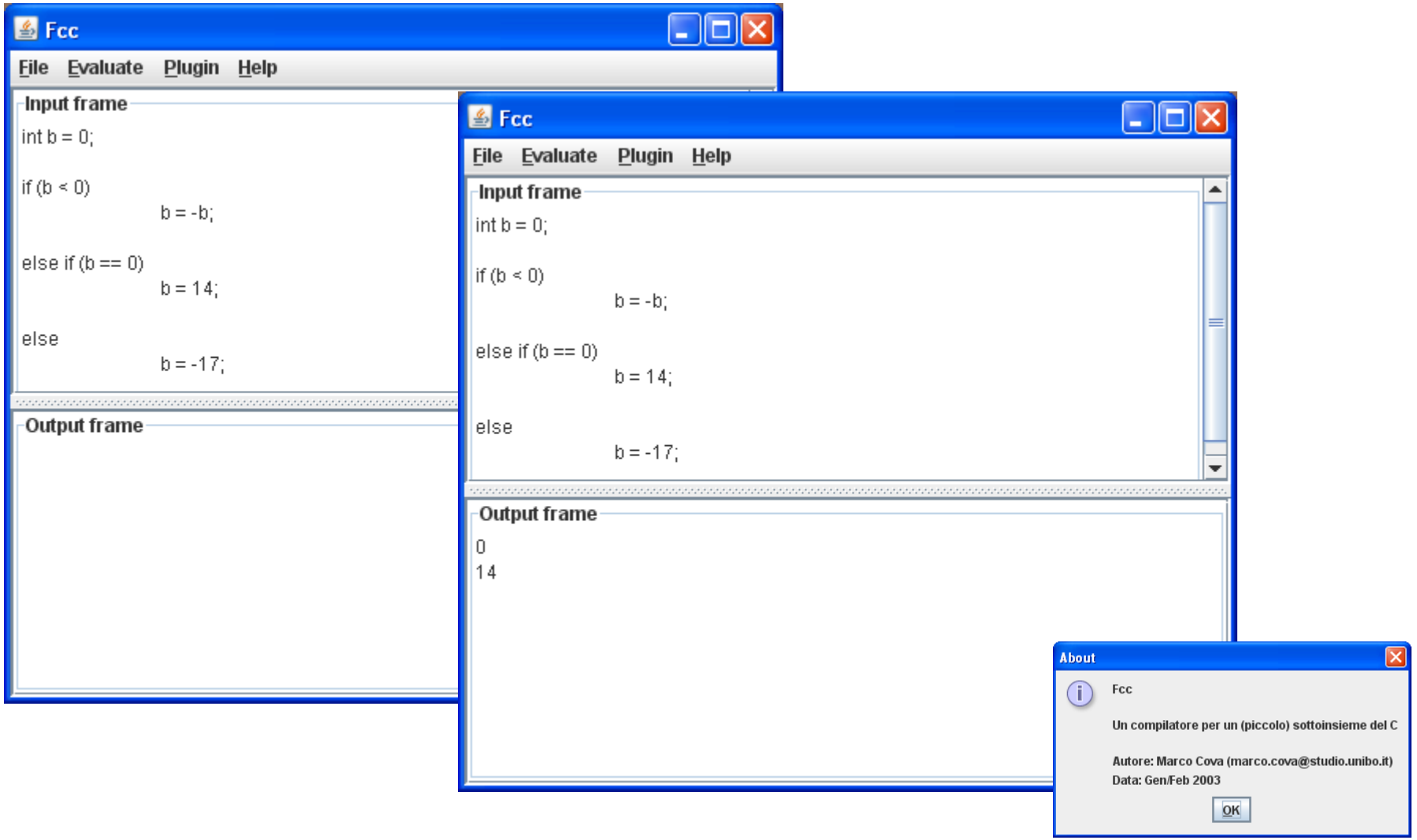
semplice if di controllo di successo/fallimento (???)



USO DA GUI (1)


Fcc
File Evaluate Plugin Help
Input frame
int a = 1;
int b = 10;
while (a < 10) {
a = a + 1;
b = b - 1;
}
b;
Output frame
Fcc
File Evaluate Plugin Help
Input frame
int a = 1;
int b = 10;
while (a < 10) {
a = a + 1;
b = b - 1;
}
b;
Output frame
1
10
2
9
3
8
4
7
5
6
About
Fcc
Un compilatore per un (piccolo) sottoinsieme del C
Autore: Marco Cova (marco.cova@studio.unibo.it)
Data: Gen/Feb 2003
OK

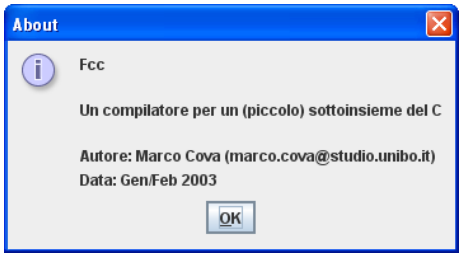
USO DA GUI (2)


Fcc
File Evaluate Plugin Help
Input frame
int b = 0;
if (b < 0)
b = -b;
else if (b == 0)
b = 14;
else
b = -17;
Output frame
Fcc
File Evaluate Plugin Help
Input frame
int b = 0;
if (b < 0)
b = -b;
else if (b == 0)
b = 14;
else
b = -17;
Output frame
0
14
About
Fcc
Un compilatore per un (piccolo) sottoinsieme del C
Autore: Marco Cova (marco.cova@studio.unibo.it)
Data: Gen/Feb 2003
OK

ESEMPIO: CALCOLO DI e = Σ 1/k!

- Dato che sono disponibili solo gli interi, calcoleremo in realtà il valore di $e * 10^7$
- Programma in SmallC:

```
int k = 0;
int kmax = 10;
int kfact = 1;
int somma= 0;
while (k < kmax) {
 somma = somma + 10000000/kfact;
 k = k + 1;
 kfact = kfact * k;
}
```



ESEMPIO: CALCOLO DI e = $\sum$ 1/k!