Laboratory of Computer Science Education

Ruolo delle variabili e pattern elementari

"I do not even know where to start"

Alberto Montresor

Università di Trento

2020/04/17

Introduzione

Soloway and Ehrlich

What is that expert programmers know that novice programmers' don't? We would suggest that the former have at least two types of knowledge that the letter typically do not:

- Programming plan: Program fragments that represent stereotypic action sequences in programming (e.g. item search loop plan)
- Rules of programming discourse: rules that specify the conventions in programming

Introduzione

Two examples of research into new concepts that can be utilized in teaching elementary programming

- Software design patterns represent language and application independent solutions to commonly occurring design problems.
  - The number of patterns is potentially unlimited;
  - There are sets of patterns
    - for various levels of programming expertise (e.g., elementary patterns for novice programmers)
    - for application areas (e.g., data structures)
- Roles of variables describe stereotypic uses of variables that occur in programs over and over again

Ruolo delle variabili e tacit knowledge

Variable plans and roles are tacit knowledge that is not mentioned explicitly in teaching programming to novices.

Expert programmers possess schemas, abstractions of concrete experiences, which help them solve programming problems and lessen the load on their working memory during problem solving.

Possession of schemas is a key difference between novices and experts, which is why instructors need to help students construct them

Insegnamento "tradizionale"

Teachers tend to present programming language constructs and students have to acquire higher-level program constructs from example programs and program fractions.

While we claim that our overall goal is to teach problem solving using a computer and a programming language, all too often learning gets bogged down into the minutiae of the syntax, semantics and pragmatics of writing a program.

Cosa sono i ruoli

Our roles are based on the nature of the successive values a variable obtains, and we pay no attention to the way the values are further used.

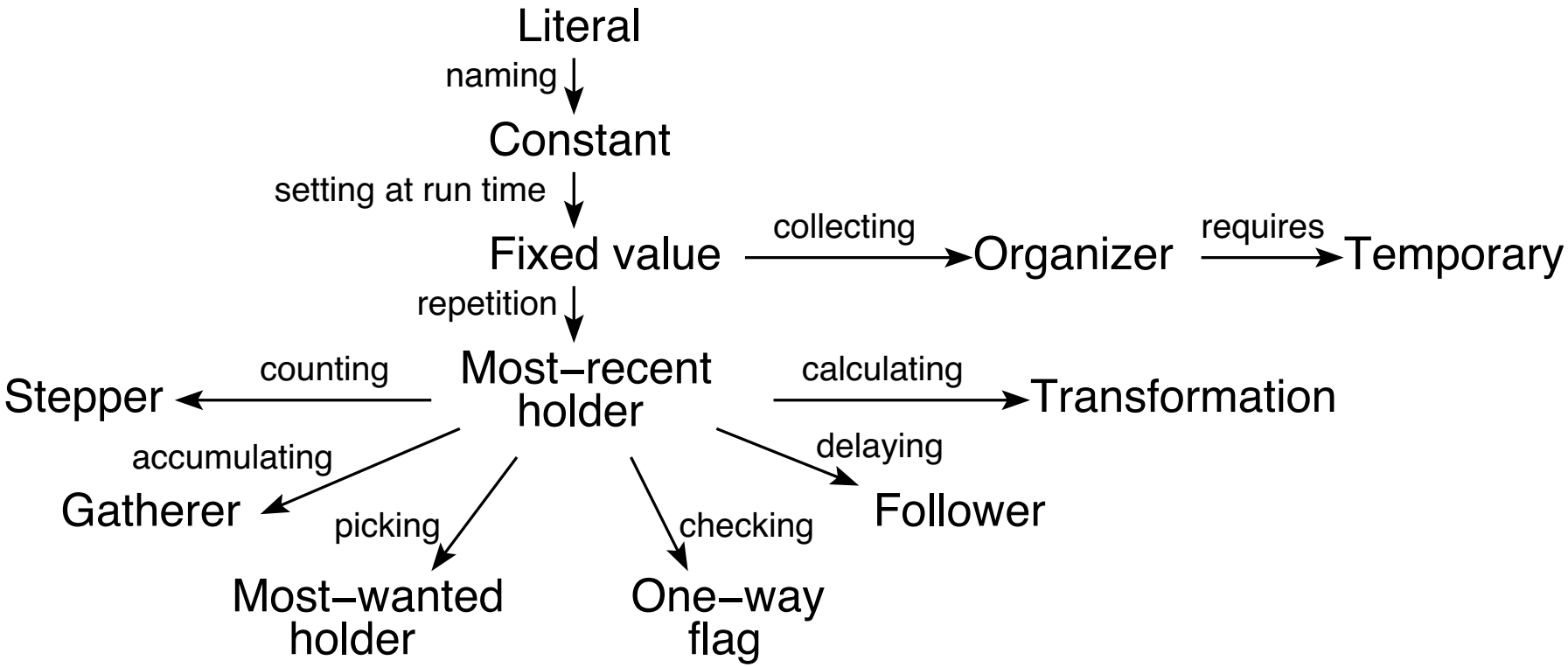
Ruolio delle variabili

Role	Informal description
Fixed-value	A variable which is assigned a value only once and the value of which does not change after that. (Sometimes known as a "single assignment" variable.)
Stepper	A variable stepping through a systematic, predictable succession of values.
Most-recent-holder	A variable holding the latest value encountered in going through a succession of unpredictable values, or simply the latest value obtained as input.
Most-wanted-holder	A variable holding the best or otherwise most appropriate value encountered so far. (For example the biggest value so far.)
Gatherer	A variable accumulating the effect of a series of individual values, for example a running-total. (Often known as an "accumulator".)
Follower	A variable that always takes its current value from the previous value of some other variable.

Ruolio delle variabili

One-way-flag	A Boolean variable that is initialised to one value and may be changed to the other value, but never reverts to its initial value once it has been changed.
Temporary	A variable holding some value for a very short time only. (For example while values are rearranged between other variables.)
Organizer	A data structure storing elements so that they can be rearranged.
Container	A data structure storing elements that can be added and removed.
Walker	A variable used to traverse a data structure, for example a pointer running down a linked list.

Relazioni dei ruoli



Quando/come insegnarli

While teaching, each role can be introduced when it is encountered for the first time in some example program during lectures.

Roles can be taught in introductory courses naturally alongside other variable-related concepts, such as type and scope

Ruoli come concetti primitivi

Roles are a cognitive concept which means that different people may assign different roles to the same variable.

Even in those cases where the assignment of a role is controversial, the debate itself can be an excellent pedagogical tool for clarifying the structure of programs in intro- ductory courses

It is important to emphasize that we do not regard roles as an end in themselves and we do not think that students should be graded on their ability to assign roles

Problemi

Misleading names

The name given to a variable can be very misleading if it is not consistent with the way the variable is used

```
temp = 0;
for (i = 0; i < key_length; i++)
   temp = ( (32*temp) + value(k[i]) ) modulo N;
```

Problemi

Variables with dual responsibilities

As we looked through our CS2 teaching materials, it turned out that some variables in our examples had ambiguous behavior

```
1  node mergesort(node list; int N) {
2
3    node first,
4         last;
5    int i;
6    if (list->next == NULL) {
7      return list;
8    } else {
9      first = list;
10     // locate middle point
11     for (i = 2; i <= (N / 2); i++)
12       list = list->next;
13     last = list->next;
14     list->next = NULL;
15     return merge(
16       mergesort(first, N / 2),
17       mergesort(last, N-(N/2)));
18   }
19 }
```

(a) before

```
1  node mergesort(node first; int N) {
2      /* first: walker, N: stepper */
3    node curr;    /* curr: walker    */
4    node mid;     /* mid: temporary */
5    int i;        /* i: stepper      */
6    if (first->next == z) {
7      return first;
8    } else {
9      curr = first;
10     // locate middle point
11     for (i = 2; i < (N / 2); i++)
12       curr = curr->next;
13    mid = curr->next;
14    curr->next = NULL;
15    return merge(
16      mergesort(first, N / 2),
17      mergesort(mid, N-(N/2)));
18    }
19  }
```

(b) after

Punti di attenzione

- We can not discard out of hand the argument that learning roles of variables adds to students' cognitive load as they have to struggle with even more variable-related terminology.
- However, we would like to stress that when we used roles to teach variable use, we did not teach our students to do anything with variables that we would not otherwise have taught. Roles just provided us with a vocabulary to better discuss these topics.
- We find it unlikely that any cognitive load inherent in learning roles would not be compensated for by the way roles clarify program behavior

Punti di attenzione

Some participants complained about "non-standard" terminology:

- in object-oriented programming "iterator" is the common name for many walkers
- in functional programming "accumulator" is used for gatherer.

We hope that a common role terminology could unify terminology in different paradigms.

Pattern

- Name Use Patterns
- Reading Data Pattern
- Read-Process-Write
- Cumulative Result Patterns
- Conversion Patterns
- Indirect Reference Patterns

Pattern

- Selection Pattern
- Repetition Patterns
  - counting - including increments other than one
  - conditioned repetition (usually a while loop)
  - polled loop (busy wait for external event - e.g. mouse click)
  - repetition with exits (usually via break or continue statement)
- Traversal patterns
  - simple linear traversal (vectors, arrays, strings)
  - streamed traversal – 'end- of-input' indicator
  - linked traversal - traversal of a linked structure
  - iterator based traversal

Index Process All Items

```
for(int k=0; k < v.size(); k++)
{
    process v[k]
}
```

Iterator Process All Items

```
Hashtable table = new Hashtable();
    // code to put values in table

    Enumeration e = table.keys();
    while (e.hasMoreElements())
    {
        process(table.get(e.nextElement()));
    }
```

Guarded linear search

```
int i=0;
boolean found = false
while(i < students.size() && !found){
  found = (student[i].grade() == 30);
}
if (found) {
  // process student[i], who has got 30
} else {
  // handle the exception
}
```

Loop and a Half

```
read value
while (value != sentinel) {
    process value
    read value
}
```

Polling Loop

```
cout << "Enter a grade between 0 and 100, inclusive: ";
cin  >> grade;

while (grade < 0 || grade > 100) {
    cout << "Sorry!  That is an illegal value." << endl;
    cout << "Enter a grade between 0 and 100, inclusive: ";
    cin  >> grade;
}
```