Concepte fundamentale ale limbajelor de programare

Structuri de control

Curs 11

conf. dr. ing. Ciprian-Bogdan Chirila

Universitatea Politehnica Timisoara
Departamentul de Calculatoare si Tehnologia Informatiei

8 mai 2023

Structuri de control

Ne concentram pe mecanisme care permit programatorului sa controleze fluxul de executie la:

- nivel de instructiune
- nivel de subprogram

Cuprins

Structuri de control la nivel de instructiune

- specifica ordinea in care instructiunile unui program individual sunt executate
- sunt grupate in trei categorii
  - Secventa
  - Selectia
  - Repetitia

Secventa

- este cea mai simpla structura de control
- este specifica limbajelor imperative
- nu ne referim la limbaje concurente
- instructiunile sunt executate in ordinea in care sunt scrise

Secventa

- in multe limbaje de programare seturi de instructiuni pot fi grupate impreuna pentru a forma o instructiune compusa
- in Python
  - sunt indentate cu un tab
- In C, Java, C#, JavaScript, TypeScript

```
{
  ...
}
```

Selectia

- ne permite sa selectam
  - o alternativa dintre doua sau mai multe disponibile
  - depinzand de o conditie logica
- in limbaje de tipul Algol-Pascal avem

```
if condition then
 sequence_of_instructions
else
 sequence_of_instructions
end if;
```

Selectia in Python

```
if test expression:
 statement(s)

if test expression:
 Body_of_if
else:
 Body_of_else

if test expression:
 Body_of_if
elif test expression:
 Body_of_elif
else:
 Body_of_else
```

Selectia intre alternative multiple

- case
  - like in Pascal, Ada, Algol 68
- switch
  - like in C, Java, C#, JavaScript
- selectia este bazata pe o valoare selectoare de tip scalar
- programatorul trebuie sa specifice variantele si valorile pentru care fiecare alternativa este selectata
- exista o optiune prin care se poate specifica o varianta ce va fi aleasa daca nu exista potrivire

Exemplu in Pascal

```
if mark <= 10 then
 case mark of
  1,2,3,4 : writeln('failed');
  5,6,7 : writeln('passed');
  8,9 : writeln('good');
  10 : writeln('excellent');
 end
else
 writeln('wrong mark');
```

Exemplu in Ada

```
case mark of
 when 1..4 => put_line("failed");
 when 5|6|7 => put_line("passed");
 when 8|9 => put_line("good");
 when 10 => put_line("excellent");
 when others => put_line("wrong mark");
end case;
```

Exemple in limbaje cu sintaxa C

```
switch(mark)
{
 case 1: case 2: case 3: case 4: printf("failed"); break;
 case 5: case 6: case 7: printf("passed"); break;
 case 8: case 9: printf("good"); break;
 case 10: printf("excellent"); break;
 default: printf("wrong mark"); break;
}
```

Inlocuitori in Python

```
def numbers_to_strings(argument):
 switcher =
 {
  1:"failed",2:"failed",3:"failed",4:"failed",
  5:"passed",6:"passed",7:"passed",
  8:"good",9:"good",
  10:"excellent",
 }
 return switcher.get(argument, "wrong mark")

if __name__ == "__main__":
 argument=5
 print (numbers_to_strings(argument))
```

Repetitia

- este mecanismul de baza pentru a face calcule complexe
- inseamna sa executi in mod repetat o instructiunie sau un set de instructiuni
- structurile repetitive sunt controlate
  - de conditii
  - de contoare

Structuri repetitive controlate de conditii

Structuri repetitive controlate de conditii

- repetitii cu test initial
  - Pascal: while condition do
  - C, Java, C#, JavaScript, TypeScript:
    `while(condition) instruction;`
- repetitii cu test final
  - Pascal: repeat instr_sequence until condition
  - C, Java, C#, JavaScript, TypeScript:
    `do instruction while(condition);`

Structuri repetitive controlate de contoare

- contorul avanseaza
  - de la o valoare initiala
  - la o valoare finala
- forma generala este

```
for variable:= initial_value to final_value step step_value do
 instruction
```

Structuri repetitive controlate de contoare

- in PI/I and Algol 68

```
for
 variable from initial_value
 by step_value
 to final value
 while condition do
sequence_of_instructions
```

Structuri repetitive controlate de contoare

- in limbajele cu sintaxa C

```
for(expr1; expre2; expr3)
 instruction;

expr1;
while(expr2)
{
 instruction;
 expr3;
}
```

Iesirea din repetitie

- salt neconditionat
  - goto
- instructiuni de salt specializat pentru a iesi din repetitii
  - in Ada
    - exit
  - in limbaje cu sintaxa C
    - break
- doar o repetitie, dar nu si cele inconjuratoare
  - In Ada
    - exit name
    - "name" este eticheta repetitiei inconjuratoare
  - in limbajele cu sintaxa C
    - continue

Cuprins

Structuri de control la nivel de subprograme
Subprograms

- efecte colaterale
  - modificarile provocate de un sunprogram asupra unei entitati care nu este locala subprogramului
  - sunt problematice in special in cazul functiilor
  - `v=a+f(a,b)+c;`
- pseudonime
  - un obiect poate fi referit de doua sau mai multe nume
  - poate aparea in cazul mecanismului de transfer a parametrilor prin adresa

Exemple de pseudonime in C++

```
int y;
----------------
void p(int &x)
{
 x=2*x;
 y=x+y;
};
-----------------
y=1;
p(y);
-----------------
```

Exemple de pseudonime in C++

- cand doua sau mai multe argumente sunt transmise prin adresa reprezentand acelasi obiect

```
int z;
------------------
void p(int &x,int &y)
{
 x=2*x;
 y=2*y;
}
------------------
z=3;
p(z,z);
```

Exemple de pseudonime in C++

- cand o structura sau componente a acelei structuri sunt transmise ca parametri

```
typedef float tab_t[100];
tab_t tab;
procedure pp(int &x, tab_t &tab)
{
  ...
}
--------------------
pp(tab[3],tab);
--------------------
```

Pseudonime in tablouri

- cand doua argumente sunt doua elemente ale aceluiasi tablou
- `p(t[i],t[j]);`
- pseudonimele apar daca `i==j`

Cuprins

Tratarea exceptiilor in Ada

- 5 tipuri de exceptii
- constraint_error
  - violarea limitelor unui subdomeniu
  - referirea ilegala a unui cand dintr-un articol cu variante
  - referirea unui ponter null
- numeric_error
  - arithmetical overflow
- storage_error
  - depasirea spatiului de memorie
- select_error, tasking_error
  - erori legate de concurenta

Tratarea exceptiilor in Ada

- declararea exceptiilor
  - error, end : exception;
- ridicarea exceptiilor
  - raise error;

Exemplu de tratare a exceptiilor

```
function f(x : float) return float is
negative : exception;
begin
if x<0 then
 raise negative;
else
 return 1/sqrt(x);
end if;
exception
 when numeric_error =>
  return 0; -- return 0 if x is 0
 when negative =>
  return -1; -- return -1 if x is negative
end f;
```

Exemplu de tratare a exceptiilor

```
package stack is
 error: exception;
 type stack_type(max_no:integer) is limited private;
 function pop(s:in out stack_type) return integer;
 procedure push(s:in out stack_type; x: integer);
 function top(s:stack_type) return integer;
 procedure init(s:out stack_type);
private
 type stack_type(max_no:integer) is
 record
  tab_st:array(1..max_no) of integer;
  ind:integer;
  end record;
end stack;
```

Exemplu de tratare a exceptiilor

```
package body stack is

 function empty(s:stack_type) return boolean is
 begin ... end empty;

 function overflow(s:stack_type) return boolean is
 begin ... end overflow;

 function pop(s: in out stack_type) return integer is
 begin
  if not empty(s) then
   s.ind:=s.ind-1;
    return s.tab_st(s.ind);
   else raise error;
end if;
 end pop;
```

Exemplu de tratare a exceptiilor

```
procedure push(s: in out stack_type; x:integer) is
begin
 if not overflow(s) then
  s.tab_st(s.ind):=x;
  s.ind:=s.ind+1;
 else raise error;
 end if;
end push;
```

Exemplu de tratare a exceptiilor

```
function top(s:stack_type) return integer is
begin
 if not empty(s) then
  return s.tab_st(s.ind-1);
 else
  raise error;
 end if;
end top;

function init(s: out stack_type) is
begin
 ...
end init;

begin
end stack;
```

Exemplu de tratare a exceptiilor

```
procedure st is
use stack;
stk:stack_type(100);
-----------------------
init(stk);
-----------------------
push(stk,10);
-----------------------
i:=top(stk);
-----------------------
```

Exemplu de tratare a exceptiilor

```
exception
 when error =>
  put_line("error using stack");
  while not empty(stk) loop
   put(pop(stk));
  end loop;
end st;
```

Tratarea exceptiilor in C#

- sunt similare cu C++ sau Java
- exceptiile sunt reprezentate prin clase
- Class System.Exception
- toate clasele reprezentand exceptii trebuie sa fie derivate din clasa predefinita Exception care este parte a spatiului de nume System
- clase derivate din Exception sunt
  - SystemException
    - generate de motorul de executie
  - AplicationException
    - generate de aplicatiile de program
    - pot fi derivate de programator prin crearea propriilor exceptii

Clase de exceptii derivate din SystemException

- ArrayTypeMismatchException
  - tipul atribuirii este incompatibil cu tipul tabloului
- DivideByZeroException
  - tentativa de a imparti cu zero
- IndexOutOfRangeException
  - indexul iese din limitele tabloului
- InvalidCastException
  - operatie de cast incorecta la rulare

Clase de exceptii derivate din SystemException

- OutOfMemoryException
  - memorie insuficienta pentru operatorul new de alocare
- OverflowException
  - supradepasire aritmetica
- StackOverflowException
  - depasirea a capacitatii stivei

Elemente de baza in tratarea exceptiilor

- cuvinte cheie
  - try
  - catch
  - throw
  - finally

Functionarea exceptiilor

- try
  - blocul try contine instructiuni sensibile la erori ce trebuiesc a fi verificate
- throw
  - daca o exceptie are loc atunci ea este aruncata
- catch
  - programul poate intercepta aceasta exceptie si o trateaza in acord cu cerintele aplicatiei
- exceptiile sunt lansate
  - in mod automat de motorul de executie C#
  - manual, utilizand cuvantul cheie throw

Tratarea exceptiilor

- codul ce va fi executat cand se iese din bloc trebuie pus in blocul finally
- pentru a prinde orice exceptie
- the orice tip
- clauza catch se foloseste fara parametru
- astfel, este creata o rutina universala pentru a intercepta si trata toate exceptiile

Exemplu simplu de tratare a exceptiilor

```
using System;
class Example1
{
 public static void Main()
 {
  int [] nums = new int[4];
  try
  {
   Console.WriteLine("Before exception");
   // we generate an exception of invalid index
   nums[7] = 10;
   Console.WriteLine("Message not to be printed");
  }
```

Exemplu simplu de tratare a exceptiilor

```
  catch(IndexOutOfRangeException)
  {
   // we intercept the exception
   Console.WriteLine("Index out of bounds");
  }
  Console.WriteLine("After catch");
 }
}
```

Exemplu simplu de tratare a exceptiilor

- text afisat:

```
Before exception
Index out of bounds
After catch
```

Generarea manuala a unei exceptii

```
using System;
class Example2
{
 public static void Main()
 {
  try
  {
   Console.WriteLine("Before throw");
   // launching  exception
   throw new DivideByZeroException();
  }
  catch(DivideByZeroException)
  {
   // intercepting the exception
   Console.WriteLine("Exception intercepted");
  }
  Console.WriteLine("After try/catch");
}
```

Generarea manuala a unei exceptii

- text afisat

```
Before throw
Exception intercepted
After try/catch
```

Cuprins

Bibliografie

1. Horia Ciocarlie – The programming language universe, second edition, Timisoara, 2013.
2. Carlo Ghezzi, Mehdi Jarayeri – Programming Languages, John Wiley, 1987.
3. Ellis Horrowitz – Fundamentals of programming languages, Computer Science Press, 1984.
4. Donald Knuth – The art of computer programming, 2002.