Fundamentals of Programming Languages

Control Structures

Lecture 11

sl. dr. ing. Ciprian-Bogdan Chirila

Lecture outline

- Instruction level control structures
  - Sequence
  - Selection
  - Repetition
- Subprograms
  - Side effects
  - Pseudonyms
- Exception handling
  - In Ada
  - In C#

Control structures

- Focus on mechanisms that allow the programmer to control the flow of actions
- Instruction level
- Subunit level

Instruction level control structures

- Specify the order in which the individual program instructions are executed
- Grouped in three categories
  - Sequence
  - Selection
  - Repetition

Sequence

- The most simple control sequence
- Implied by imperative languages
- We do not refer concurrent languages
- The instructions are executed in the order in which they are written

Sequence

- In many PLs set of instructions can be grouped together to form a composed instruction
- In Pascal
  - Begin
  - End
- In C
  - {
  - }

Selection

- Allow us to select
  - an alternative between two or more available
  - Depending on a logical condition
- In Algol-Pascal like PLs

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

Selection between multiple alternatives

- case
  - like in Pascal, Ada, Algol 68
- switch
  - like in C
- Selection is based on a selector value of scalar type
- The programmer must specify the variants and the values for which each value is selected

Selection between multiple alternatives

- There is an option of specifying a variant to be chosen when there is no match
- Not present in standard Pascal
- It is included in different implementations
  - like Turbo Pascal

Pascal example

```
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');
```

Ada Example

```
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 ;
```

C Example

```
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;
}
```

Repetition

- The base mechanism for making complex computations
- To execute repeatedly an instruction or a set of instructions
- Structures controlled
  - By condition
  - By counter

Condition controlled structures

- Repetition with final test
  - Pascal: while condition do
  - C: while(condition) instruction
- Repetition with initial test
  - Pascal: repeat instr_sequence until condition
  - C: do instruction while(condition)

Counter controlled structures

- The counter evolves
  - from an initial value
  - to a final value
- The general form

for variable:= initial_value to final_value

step step_value do instruction

Counter controlled structures

- In Pl/I and Algol 68

for

variable from initial_value

by step_value

to final value

while condition do

sequence_of_instructions

Counter controlled structures

- In C

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

Expr1;
while(expr2)
{
        instruction;
        expr3;
}
```

Exiting the repetition

- Unconditional jump
  - goto
- Specialized jump instruction to exit a loop
  - In Ada
    - exit
  - In C
    - break
  - Only one loop, but not all surrounding ones
  - In Ada
    - exit name
    - Name is the label of the outer loop
  - In C
    - continue