Lecture 7

Control Structures

In most programming languages there is the importance of readability and ease of programming over the internal structures of controls.

1. Selection

FORTRAN IV single-way IF vs. C's (and most other languages)

Example
C:  if ( warning == 1 )  
{  
I = 1;  
J = 2;  
}  

FORTRAN:  IF ( WARNING .NE. 1) GO TO 20  
I = 1  
J = 2  
20:  CONTINUE

Problems with nested if's
Generally the else clause in a nested if belongs to the nearest if..then clause
Consider the following statement: *"if the warning is 0 then, if the counter is 0 write 'ERROR.' Otherwise, always write 'CORRECT.'"

WRONG LOGIC:  CORRECT LOGIC
if (warning == 0 )  if (warning == 0 )  
{  
if ( counter == 0 )  
cout << "ERROR";
else  
cout << "CORRECT";
}  
else  
cout << "CORRECT";

Ada's Version     FORTRAN IV Version
if WARNING = 0 then  IF (WARNING .NE. 0 ) GO TO 20  
if COUNTER = 0  IF (COUNTER .NE. 0 ) GO TO 30  
PUT("ERROR");  PRINT *, 'ERROR'  
end if;  GO TO 35  
else  
20:  PRINT *, "CORRECT"  
PUT("CORRECT");  GO TO 35  
end if;  30:  PRINT *, "ERROR"  
35:  CONTINUE

2. Multi-way Selection
Pascal:

```pascal
case choice of
    'A', 'a' : begin
        AddResult( index1, index2 );
        count := count + 2
    end;
    'B', 'b' : begin
        MultiplyResult( index1, index2 );
        count := count + 5
    end;
else
    Writeln( 'ERROR' )
end
```

C:

```c
switch (choice ) {
    case 'A':
    case 'a':
        AddResult( index1, index 2);
        count += 2;
        break;
    case 'B':
    case 'b':
        MultiplyResult( index1, index2);
        count += 5;
        break;
    default: cout << "ERROR" << endl;
}
```

Operational Semantics Description

```c
if choice = 'A'
goto Achoice;
if choice = 'a'
goto Achoice;
if choice = 'B'
goto Bchoice;
if choice = 'b'
goto Bchoice;
cout << "ERROR" << endl;
goto Out;
Achoice:
    AddResult( index1, index2);
    count += 2;
    goto Out;
Bchoice:
    MultiplyResult( index1, index2);
```
count += 5;

Out:

Note use of Ada's \texttt{elsif}

3. Iterative Statements

for statements:

- Pascal's for: for variable := initial\_value (to \texttt{| downto}) final value do

- Ada's for: for variable in \texttt{[reverse]} discrete\_range loop
  
  ....

  end loop;

- C/C++/C#/Java's for:

  Logically controlled loops
  while, do .. while, repeat .. until, loop..end loop (with if boolean then exit and exit when boolean)
  
  C/C++'s "continue" and "break"

foreach in Perl, Visual Basic, and C#