기본 문법-조건문과 반복문

■ if 문

if 문에는 if...then과 if...then...else의 두가지 형태가 있습니다. if...then 문장의 문법은 다음과 같습니다.

```
if expression then statement
```

여기서, expression은 부울 값을 리턴하는 표현식입니다. expression이 True인 경우 statement가 실행되고 그렇지 않은 경우 실행되지 않습니다. 예를 들면,

```
if J <> 0 then Result := I/J;
```

if...then...else 문의 문법은 다음과 같습니다.

```
if expression then statement1 else statement2
```

여기서, expression은 부울 값을 리턴합니다. expression이 True인 경우 statement1이 실행되고 그렇지 않은 경우 statement2가 실행됩니다. 예를 들면,

```
if J = 0 then
  Exit
else
  Result := I / J;
```

then과 else 절은 각각 하나의 문장을 포함하지만, 이 각 문장은 구조문이 될 수도 있습니다. 예를 들면,

```
if J <> 0 then
begin
  Result := I/J;
  Count := Count + 1;
end
else if Count = Last then
  Done := True
else
  Exit;
```

then 절과 else 사이에 세미콜론이 없다는 사실에 주의하십시오.

연속으로 중첩된 if 문을 사용할 때에는 특별한 주의가 필요합니다. 일부 if 문에는 else 절이 있고, 다른 if 문에서는 else 절이 없기 때문에 이러한 문제가 발생합니다. if 문에 비해 else 절이 적은 연속된 중첩 조건문에서는, 어떤 else 절이 어떤 if에 속하는지 명확하게 보이지 않을 수도 있습니다.

```
if expression1 then if expression2 then statement1 else statement2;
```

이를 해석하는 방법에는 두가지 방법이 있을 것입니다.

```
if expression1 then [ if expression2 then statement1 else statement2 ];
if expression1 then [ if expression2 then statement1 ] else statement2 ;
```

컴파일러는 항상 첫 번째 방법으로 해석합니다. 즉, 실제 코드에서 다음 문장은,

```
if ...{ expression1 } then
  if ...{ expression2 } then
    ... { statement1 }
  else
    ... { statement2 } ;
```

다음 문장과 동일합니다.

```
if ...{ expression1 } then
begin
  if ...{ expression2 } then
    ... { statement1 }
  else
    ... { statement2 }
end;
```

중첩된 if 문을 해석할 때의 규칙은, 가장 안쪽에 있는 조건문부터 시작하여 각각의 else를 왼쪽 방향으로 가장 가까운 if로 묶으면서 해석한다는 것입니다. 컴파일러가 위의 두 번째 방식으로 이해하도록 하려면 다음과 같이 명시적으로 작성해야만 합니다.

```
if ...{ expression1 } then
begin
  if ...{ expression2 } then
    ... { statement1 }
end
else
  ... { statement2 } ;
```

[연습문제]

실수 x,y(x와y는 같지 않은 수) 가 주어지면 작은 수는 두수의 평균값으로 바꾸고 큰 수는 자신의 3배로 바꿔보자.

■ case 문

case 문은 복잡하게 중첩된 if 조건문에 비해 더 이해하기 쉬운 대안이 될 수 있습니다. case 문의 형식은 다음과 같습니다.

```
case selectorExpression of
  caseList1: statement1;
  ...
  caseListn: statementn;
end
```

여기서, selectorExpression은 서수(ordinal) 타입(문자 타입은 사용할 수 없음) 표현식이고, caseList는 다음 중 하나입니다.

- 선언된 숫자 상수이거나 컴파일러가 프로그램을 실행하지 않고 계산할 수 있는 기타 표현식.

  selectorExpression과 호환되는 서수 타입이어야 합니다. 따라서 7, True, 4 + 5 * 3

  모두 caseList로 사용할 수 있지만, 변수와 대부분의 함수 호출은 사용할 수 없습니다.

- First..Last 형식의 부분 범위(subrange). 여기서, First와 Last는 위의 조건을 모두 만족해야 하며 First는 Last보다 작거나 같아야 합니다.
- item1, ..., itemn 형식의 리스트. 여기서, 각 item은 위의 조건들 중 하나를 만족해야 합니다.

case 문의 마지막에는 else 절이 올 수 있습니다.

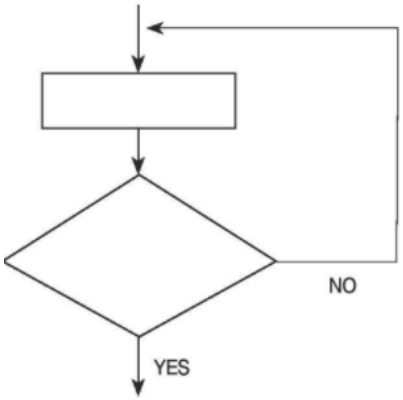
```
case selectorExpression of
  caseList1: statement1;
  ...
  caseListn: statementn;
else
  statements;
end
```

■ 순환문

순환문(루프, loop)는 언제 실행을 중단시킬지를 결정할 제어 조건이나 변수를 이용하여 일련의 문장들을 반복적으로 실행할 수 있게 해줍니다. 델파이에는 repeat 문, while 문 및 for 문의 세가지 제어 루프(control loop)가 있습니다.

표준 Break와 Continue 프로시저를 사용하여 repeat, while, for 문의 흐름을 제어할 수 있습니다. Break는 Break가 있는 곳에서 문장을 종료하며, Continue는 순환문 내에서 그 이후의 문장들을 무시하고 순환문의 다음 반복을 실행합니다.

■ repeat 문



Repeat··· Until 플로우차트

repeat 문의 문법 형식은 다음과 같습니다.

```
repeat statement1; ...; statementn; until expression
```

여기서, expression은 부울 값을 리턴합니다. until 앞에 있는 마지막 세미콜론(;)은 옵션입니다. repeat 문은 repeat 문 내에 있는 문장들을 차례로 실행하고, 매번 반복이 끝난 후에 expression을 테스트합니다. expression이 True를 리턴하면 repeat 문은 종료됩니다. 첫번

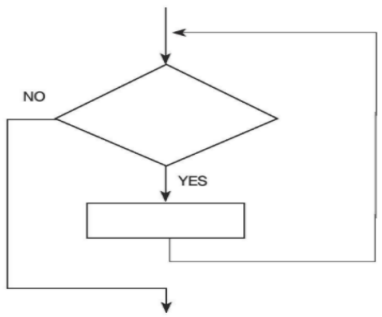
째 반복이 끝나기 전까지는 expression을 계산하지 않기 때문에 repeat 내에 있는 문장들은 최소한 한 번 이상 실행됩니다.

repeat 문의 예제는 다음과 같습니다.

```
repeat
  K := I mod J;
  I := J;
  J := K;
until J = 0;

repeat
  Write('Enter a value (0..9): ');
  Readln(I);
until (I >= 0) and (I <= 9);
```

■ while 문



while…do 플로우차트

while 문은 repeat 문과 유사하지만, while 문 내에 있는 문장들을 실행하기 전에 제어 조건을 계산한다는 점이 다릅니다. 따라서 조건이 False면 while 문 내에 있는 문장들은 전혀 실행되지 않습니다.

while 문의 문법 형식은 다음과 같습니다.

```
while expression do statement
```

```
while Data[I] <> X do I := I + 1

while I > 0 do
begin
  if Odd(I) then Z := Z * X;
  I := I div 2;
  X := Sqr(X);
end

while not Eof(InputFile) do
begin
  Readln(InputFile, Line);
  Process(Line);
end;
```

■ **for** 문

repeat 또는 while 문과 달리, for 문은 루프의 반복 횟수를 명시적으로 지정해야 합니다.

for 문의 문법 형태는 다음과 같습니다.

```
for counter := initialValue to finalValue do statement
```

```
for counter := initialValue downto finalValue do statement
```

for 문의 예는 다음과 같습니다.

```
for I := 2 to 63 do
  if Data[I] > Max then
    Max := Data[I];

for I := ListBox1.Items.Count - 1 downto 0 do
  ListBox1.Items[I] := UpperCase(ListBox1.Items[I]);

for I := 1 to 10 do
  for I := 1 to 10 do
  begin
    X := 0;
    for I := 1 to 10 do
    X := X + Mat1[I, K] * Mat2[K, J];
    Mat[I, J] := X;
  end;

for C := Red to Blue do Check(C);
```