This simple search statement works only if one is sure that there is at least one person with `Height` equal to `175` on the list. But is this realistic? A check for failing to find `175` before reaching the end of the list is mandatory unless you can guarantee it. We might first try the following solution:

```
Pt := First;
while (Pt <> nil) and (Pt↑.Height <> 175) do
   Pt := Pt↑.Next
```

But recall Section 4.A. If `Pt = nil`, the variable `Pt↑`, referenced in the second factor of the termination condition, *does not exist* at all, and referencing it is an error. The following are two possible solutions which treat this situation correctly:

```
(1)  Pt:= First; B := true
     while (Pt <> nil) and B do
        if Pt↑.Height = 175 then B := false
        else Pt := Pt↑.Next

(2)  Pt := First;
     while Pt <> nil do
        begin if Pt↑.Height = 175 then goto 13;
           Pt := Pt↑.Next
        end;
     13:
```

10.B. New and Dispose

To pose another problem, say we wish to add the sample person to the data base. First a variable must be allocated, and its identifying value obtained by means of the predeclared procedure `New`.

`New(P)` a procedure that allocates a new identified (dynamic) variable `P↑` having as its type the domain type of `P`, creates a new identifying pointer value having the type of `P`, and assigns it to `P`. If `P↑` is a variant record, `New(P)` allocates enough space to accommodate all variants.

`New(P,C1,...,Cn)` allocates a new identified (dynamic) variable `P↑` having the variant record type of `P` with tag field values `C1,...,Cn` for `n` nested variant parts, creates a new identifying pointer value having the type of `P`, and assigns it to `P`.