# 5.4 Error Checking

Beginning programmers often get frustrated when writing programs and there are many syntax errors. But those type of errors are easier to fix because Turing tells you something is wrong (sometimes exactly what is wrong) and where (or at least close) the problem is.

Errors that are harder to find and fix are ones that occur when the program is running. Ones that crash your program, though annoying, are easier to find and fix. When the program crashes you know something is wrong and usually the line that caused the problem is highlighted. More difficult to find are ones where the program runs but produces incorrect results. It can take some time to pinpoint where the problem is.

To find errors with arrays there are two places where things can go wrong. The most common problem is an index that is out of bounds. For example if you had an array:

```
var a : array 1 .. 5 of int
```

and you tried to do: `a(6) := 3`

then the program would crash since index values can only go from 1 to 5.

The other thing to be careful of is to make sure the element you are trying to store in the array is the correct type.

To avoid problems with the index, it is best to use a for loop to travel through the array when possible. Then you don't have to worry about using the wrong type or having a index out of range.

Example: Search for a value in an array.

```
var a : array 1 .. 4 of int := init(10, 20, 30, 20)
var location : int := 0 % if we don't find it location will still be 0
var find : int := 35
for i : 1 .. 4
   if a(i) = find then
      location := i
   end if
end for
```

By using a `for` loop we don't have to worry about the index being out of range. If `find` was equal to 20 location would be 4 since we would first make location equal to 2 but then we'd find it again at index 4. If we wanted to stop looking as soon as we find it, we'd have to use a `loop` statement instead. Then we'd have to be more careful to make sure the index stays in range.

```
var i : int := 0
var location : int := 0
var find : int := 20
loop
   i := i + 1
   if a(i) = find then
      location := i
      exit  % like exit when but exits the loop no matter what
   end if
   exit when i = 4 % will crash otherwise if we don't find the value
end loop
```

In this program `location` will be equal to 2 since we exit the loop when we find what we are looking for.

If the user can enter the index, then you have to be careful in case they enter something invalid. Suppose you had a program where you were tracking the price of something from the year 2000 to the year 2040. You could declare an array like this:

```
var price: array 2000 .. 2040 of real
```

Then suppose later in the program you wanted to ask the user what year they wanted to see the price for:

```
var year: int
put "What year do you want?"
get year
put "The price in ", year, " was $", price(year)
```

If they user types 1990 for the year this program will crash, since 1990 is an invalid index. Instead you should use an if statement after you get the year, to make sure the index is valid:

```
if year < 2000 or year > 2040 then
   put "Invalid year"
else
   put "The price in ", year, " was $", price(year)
end if
```

Examples of different types of errors:

```
var values : array 1 . 10 of int
```
Syntax error - needs 2 dots

```
var nums : array 1 .. 100 of int
nums(15) := 5.3
```
Wrong type to store in array

```
nums(200) := 11
```
Index out of range

---

Read section 5.4 in the textbook. Do exercise 5.4 #1-5 (I'll post solutions on Friday). Remember the assignment is due Monday. We will start chapter 6 on Tuesday.