

How to **THINK** like a Programmer

Problem Solving for the Bewildered

paul vickers



Problem Solving 6

Indeterminate Loops

Aim



- ▶ In this lesson we will begin our study of looping control structures
 - ▶ indeterminate and determinate loops
 - ▶ why we need loops
 - ▶ the `while` loop construct
 - ▶ tackling a problem involving a `while` loop in an indeterminate scenario in detail

Determinate and Indeterminate loops



- ▶ Iteration (looping) structures come in two flavours:
 - ◉ **Determinate** — one in which the number of times the action block is to be repeated is known in advance or can be calculated at the time the loop is executed
 - ◉ **Indeterminate** — one in which the number of iterations is unknown in advance and cannot be calculated

Why use loops?



- Imagine the problem of making some tea - what if more than 1 spoon of sugar is required? We need to repeat the task of adding sugar.

- If two sugars are required we *could* do this:

```
if (sugar required)
    Add spoonful of sugar
endif
if (more sugar required)
    Add spoonful of sugar
endif
```

- But what is wrong with this approach?
 - ◉ It is **not very efficient** to write it this way, what if there were 5 spoons of sugar required? There is an action here that needs to be repeated several times.

The `while` construct



- A better approach is to use another construct called `while`

- Now we can do this:

```
while (more sugar required)
    Add spoonful of sugar
endwhile
```

- The general form of the while structure in pseudocode is

```
while (condition is true)
```

```
    Action 1
```

```
    Action 2
```

```
    ...
```

```
    Action n
```

```
endwhile
```

```
next action
```

The action block executes as long as the loop condition is true. When it becomes false the loop stops and the action following the loop is executed.

Indeterminate iterations



- ▶ Here the **number of loop iterations cannot be determined in advance**
- ▶ Imagine a situation where it is required to calculate the average age of a class of children. The algorithm should be able to calculate the average for any number of children (including zero). Children's ages will be entered one at a time. When there are no more ages to enter the user will indicate this by entering an age value of zero. Once all the ages have been entered the algorithm will display the average age of the group to 1 decimal place.

Indeterminate iterations - tackling a problem



- ▶ Note that this problem statement tells us a few important things.
 - ▶ We know that we have to read in the ages of a group of children of unknown size, one at a time.
 - ▶ We also know that once we have finished entering the ages, we must signal this by entering a zero value for the age.
 - ▶ We also know that once we have entered all the ages, then the algorithm needs to calculate the average age of the children.
 - ▶ We know that we need to display the average age to 1 decimal place
- ▶ A first draft of pseudocode for the solution here is:

```
1 Get all the ages from the user
2 Calculate and display the average age to 1 decimal
place.
```

Indeterminate iterations

- tackling a problem



- ▶ Getting the childrens ages from the user is a repetitive process so we should use a loop for this.
- ▶ A second draft of pseudocode for the solution here is:
 - 1 Get the first age from the user
 - 2 while (there are more ages to be processed)
 - 2.1 process the age
 - 2.2 get the next age from the user
 - endwhile
 - 3 Calculate and display the average age to 1 decimal place.
- ▶ This looks okay so far but **some parts are too vague** such as **there are more ages to be processed** and **process the age**

Indeterminate iterations

- tackling a problem



- ▶ Recall that the loop should stop when the **user enters the value zero for the age**. This is the condition which stops the loop. Therefore the condition **there are more ages to be processed** can be replaced with **the value entered for age is not zero** or even better, **age \neq zero**
- ▶ What do we mean by **process the age**? Recall that we need to determine the average age of the children. This is only possible provided we know the total age of all the children combined. Therefore, the processing of the age here really means keeping a running total of the ages as they are entered by the user. So, **process the age** can be replaced with **keep a running total of the ages**

Indeterminate iterations

- tackling a problem



- ▶ A third draft of pseudocode for the solution here is:

```
1 Get the first age from the user
2 while (age ≠ zero)
    2.1 keep a running total of the ages
    2.2 get the next age from the user
endwhile
3 calculate and display the average age to 1 decimal
   place.
```

- ▶ Now the processes of **keeping a running total of the ages** and **calculating the average** need to be considered.
- ▶ Keeping a running total of the ages means that we need to refer to a variable in our algorithm called *totalAge*. Initially, (before the loop begins) we **set the value of this variable to zero**. Then, with each loop iteration, the value of *totalAge* gets increased by the current age value entered.

Indeterminate iterations - tackling a problem



- ▶ A fourth draft of pseudocode for the solution here is:

```
1 Initialise totalAge to zero
2 Get the first age from the user
3 while (age ≠ zero)
    3.1 set totalAge to totalAge + age
    3.2 get the next age from the user
endwhile
4 Calculate and display the average age to 1 decimal place.
```

- ▶ Now we turn our attention to calculating the **average** of the ages. This is a relatively simple operation that just involves **dividing the final value of the variable totalAge by the number of children in the group.**
- ▶ We have already dealt with *totalAge* but the number of children in the group is something we have not considered.

Indeterminate iterations

- tackling a problem



- ▶ To keep track of the total number of children that were in the group we need to refer to another variable in our algorithm. We can call it *numberOfChildren*. Like, *totalAge*, this needs to be **initialised to zero** before the loop begins. Then, as the loop iterates, it will be increased by one each time. In other words, every time we enter another age, we are processing another child, so *numberOfChildren* needs to get updated.
- ▶ A fifth draft of pseudocode for the solution here is:
 1. Initialise *totalAge* to zero
 2. Initialise *numberOfChildren* to zero
 3. Get the first age from the user
 4. while (age \neq zero)
 - 4.1 set *totalAge* to *totalAge* + age
 - 4.2 set *numberOfChildren* to *numberOfChildren* + 1
 - 4.3 get the next age from the user
 - endwhile

Indeterminate iterations - tackling a problem



5. Set the average to $\text{totalAge} \div \text{numberOfChildren}$
6. Display the average to 1 decimal place

- ▶ Our algorithm looks okay. If we want to be critical about it, maybe instead of using the words **Get the first age from the user**, we could have 2 separate actions here that say

Prompt the user for a child's age
Read in the child's age

- ▶ More important than this, however, is to **consider the person running the program** ultimately. Imagine that they have no prior knowledge of the software. When they are running the program, how will they know how to exit the loop? The answer is **they won't** – so you need to tell them what value is used to bring the loop to a halt – this value is called the **sentinel value**. In fact, this type of loop scenario is called a **data-sentinel controlled while loop**.

ACTIVITY

Write the sixth draft of the pseudocode now for this problem.

There is at least one important omission from our algorithm that is not very obvious. Can you spot it?

Hint: it's a **runtime** issue

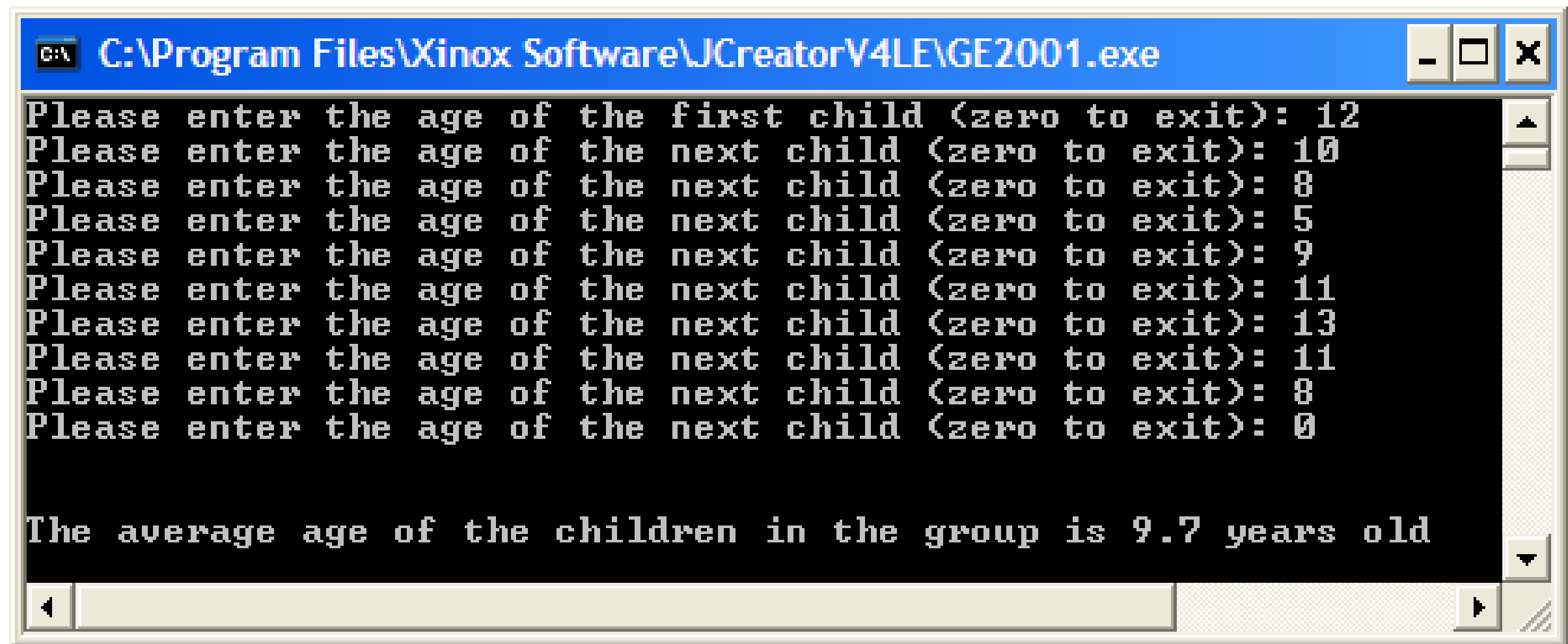
Solution



Pseudocode solution to determine the average age of a group of children of unknown size

1. Initialise totalAge to 0
2. Initialise numberOfChildren to 0
3. Prompt the user for a child's age, indicating the value of zero to stop input.
4. Read in the child's age
5. while (age \neq 0)
 - 5.1 set totalAge to totalAge + age
 - 5.2 set numberOfChildren to numberOfChildren + 1
 - 5.3 Prompt the user for a child's age, indicating the value of zero to stop input.
 - 5.4 Read in the child's ageendwhile
6. if (numberOfChildren > 0)
 - 6.1 set the average to totalAge \div numberOfChildren
 - 6.2 display the average to 1 decimal placeendif

Now that you have the final pseudocode solution to the problem, you should try to translate it to a working Java program that solves the problem. Your program might run as indicated in the following sample screenshot:



```
C:\Program Files\Xinox Software\JCreatorV4LE\GE2001.exe

Please enter the age of the first child (zero to exit): 12
Please enter the age of the next child (zero to exit): 10
Please enter the age of the next child (zero to exit): 8
Please enter the age of the next child (zero to exit): 5
Please enter the age of the next child (zero to exit): 9
Please enter the age of the next child (zero to exit): 11
Please enter the age of the next child (zero to exit): 13
Please enter the age of the next child (zero to exit): 11
Please enter the age of the next child (zero to exit): 8
Please enter the age of the next child (zero to exit): 0

The average age of the children in the group is 9.7 years old
```