Algorithms, Part 2 of 3

Topics

- Problem Solving Examples
- Pseudocode
- Control Structures

Problem Solving

- Decode this sentence:

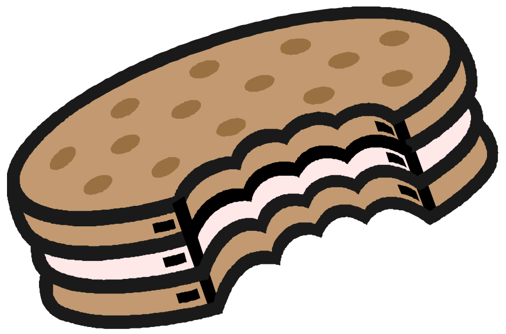
  Pdeo eo pda yknnayp wjosan.

- We have just come up with a specific solution to a problem.
- Can this solution be generalized?

Problem Solving (con’t)

- Now that we know what algorithms are, we are going to try some problem solving and write algorithms for the problems.
- We’ll start with step-by-step instructions that solve a particular problem and then write a generic algorithm that will solve any problem of that type.

Someone Stole a Cookie from the Cookie Jar



Problem: Momma had just filled the cookie jar when the 3 children went to bed. That night one child woke up, ate half of the cookies and went back to bed. Later, the second child woke up, ate half of the remaining cookies, and went back to bed. Still later, the third child woke up, ate half of the remaining cookies, leaving 3 cookies in the jar. How many cookies were in the jar to begin with?

Specific Solution to the Cookie Problem

- First, we solve the specific problem to help us identify the steps.
  - 3 cookies left X 2 = 6 cookies left after 2nd child
  - 6 X 2 = 12 cookies left after 1st child
  - 12 X 2 = 24 = original number of cookies

A Generic Algorithm

- What is a **generic algorithm** for this problem?

  An algorithm that will work with <u>any number of remaining cookies</u>

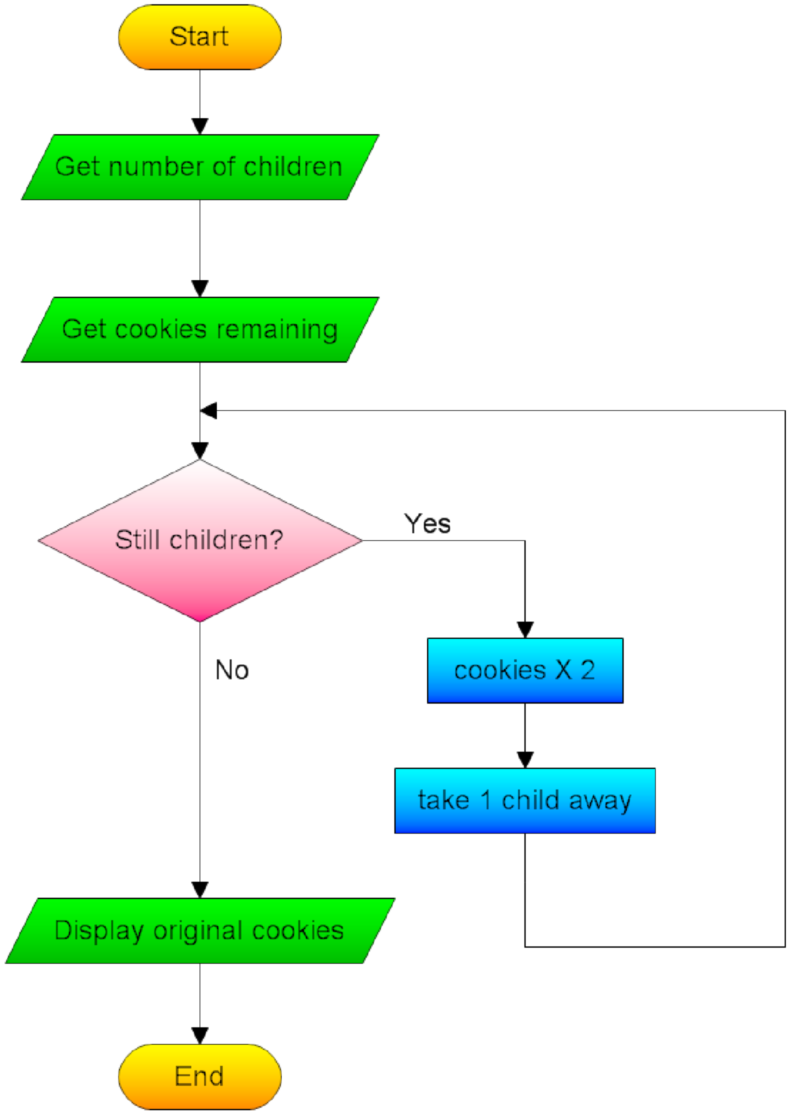
  AND

  that will work with <u>any number of children</u>.

Generic Algorithm for Cookie Problem

- Get number of children.
- Get number of cookies remaining.
- While there are still children that have not raided the cookie jar, multiply the number of cookies by 2 and reduce the number of children by 1.
- Display the original number of cookies.

Flowchart for Cookie Problem



Pseudocode

- When we broke down the previous problem into steps, we expressed each step as an English phrase.
- We can think of this as writing **pseudocode** for the problem.
- Typically, pseudocode is a combination of English phrases and formulas.

Pseudocode (con't)

- Pseudocode is used in
  - designing algorithms
  - communicating an algorithm to the customer
  - converting an algorithm to code (used by the programmer)
  - **debugging** logic (semantic) errors in a solution before coding (**hand tracing**)
- Let's write the Cookie Problem algorithm using a more formal pseudocode and being more precise.

Improved Pseudocode

```
Display "Enter the number of children:  "
Read <number of children>
Display "Enter the number of cookies remaining:  "
Read <cookies>
While (<number of children>  >  0)
   <cookies> = <cookies> X 2
   <number of children> = <number of children> - 1
End_While
Display "Original number of cookies = ", <cookies>
```

Alternative Pseudocode

```
Display "Enter the number of children:  "
Read <number of children>
Display "Enter the number of cookies remaining:  "
Read <cookies >
<counter> = 1
While (<counter>  <=  <number of children>)
    <cookies> = <cookies> X 2
    <counter> = <counter> + 1
End_While
Display "Original number of cookies = ", <cookies>
```

Observations

- Any **user prompts** should appear exactly as you wish the programmer to code them.
- The destination of any output data should be stated, such as in "Display", which implies the screen.
- Make the data items clear (e.g., surround them by < and >) and give them descriptive names.
- Use formulas wherever possible for clarity and brevity.
- Use keywords (such as Read and While) and use them consistently. Accent them in some manner.

Observations (con’t)

- Use indentation for clarity of logic.
- Avoid using code. Pseudocode should not be programming language-specific.
- Always keep in mind that you may not be the person translating your pseudocode into programming language code. It must, therefore, be unambiguous.
- You may make up your own pseudocode guidelines, but you MUST be consistent.

Brian’s Shopping Trip

Problem: Brian bought a belt for $9 and a shirt that cost 4 times as much as the belt. He then had $10. How much money did Brian have before he bought the belt and shirt?

Specific Solution to Shopping Problem

Start\$ = Belt\$ + Shirt\$ + \$10

Start\$ = Belt\$ + (4 X Belt\$) + \$10

Start\$ = 9 + (4 X 9) + 10 = \$55

Generic Algorithm for Shopping Problem

- Now, let's write a generic algorithm to solve any problem of this type.
- What are the inputs to the algorithm?
  - the cost of the first item (doesn't matter that it's a belt): <item1 price>
  - the number to multiply the cost of the first item by to get the cost of the second item: <multiplier>
  - the amount of money left at the end of shopping: <amount left>

Generic Algorithm for Shopping Problem (con’t)

- What are the outputs from the algorithm?
  - the amount of money available at the start of the shopping trip: <start amount>
- Note that we may end up needing some intermediate variables.

Pseudocode

```
Display "Enter the price of the first item:  "
Read <item 1 price>
Display "Enter the multiplier:  "
Read <multiplier>
Display "Enter the amount left after shopping:  "
Read <amount left>
<item2 price> = <multiplier> X <item1 price>
<start amount> = <item1 price> + <item2 price> +
                         <amount left>
Display "The starting amount was ", <start amount>
```

Control Structures

Any problem can be solved using only three logical **control structures**:

- Sequence
- Selection
- Repetition

Sequence

- A series of steps or statements that are executed in the order they are written.
- Example:

```
Display "Enter a number:  "
Read <number1>
Display "Enter another number:  "
Read <number2>
<sum> = <number1> + <number2>
Display "sum = ", <sum>
```

Selection

- Defines one or more courses of action depending on the evaluation of a condition.
- Synonyms: **conditional**, **branching**, **decision**
- Examples:

```
If (<age> >= 18)
   Display "Go vote!"
End_if
```

```
If (<age> >= 18)
   Display "Go vote!"
Else
   Display "Maybe next time!"
End_if
```

Repetition

- Allows one or more statements to be repeated as long as a given condition is true.
- Synonyms: **looping**, **iteration**
- Example:

```
While (condition is true)
    do this
End_while
```

- Notice the repetition structure in the Cookie Problem pseudocode.