Elements of Algorithms

Mark Maloof
Department of Computer Science
Washington, DC  20057

September 29, 2015
Four Critical Elements

- Algorithms have some subset of the following critical elements:
  1. simple statements, including but not limited to:
     - input statements
     - assignment statements
     - output statements
     - return statements
  2. sequences of statements, which are also statements
  3. branching statements
  4. looping statements
Algorithm for Simple Interest

1: **input** $r$, $b$, and $m$  ▶ input statement
2: $i \leftarrow r \cdot b \cdot m$   ▶ assignment statement
3: **output** $i$                   ▶ output statement
Another Algorithm for Simple Interest

1: input $r$
2: input $b$
3: input $m$
4: $i \leftarrow r$
5: $i \leftarrow i \cdot b$
6: $i \leftarrow i \cdot m$
7: output $i$
Branching: If-then Statement

```plaintext
if  some condition is true  then
    statement (or sequence)
end if
```
Flowchart for an if-then Statement
Example of an if-then Statement

1: **input** grade
2: **if** grade > 64  **then**
3:    **output** pass
4: **end if**
5: **if** grade ≤ 64  **then**
6:    **output** fail
7: **end if**
Branching: If-then-else Statement

if some condition is true then
    statement (or sequence)
else
    statement (or sequence)
end if
Flowchart for an if-then-else Statement
Example of an if-then-else Statement

1: input grade
2: if grade > 64 then
3: output pass
4: else
5: output fail
6: end if
Looping: While Statement, While Loop

```plaintext
while some condition is true do
  statement (or sequence)
end while
```
Flowchart for a While Loop

- Condition
- Statement
- True
- False

The flowchart illustrates the control flow in a while loop, where the loop continues as long as the condition is true.
Example of a While Loop

1: input grade
2: while there is a grade do
3:   if grade > 64 then
4:     output pass
5:   else
6:     output fail
7:   end if
8: end while
repeat
  statement (or sequence)
until some condition is true

Equivalent to:

statement (or sequence)
while some condition is false do
  statement (or sequence)
end while
For Loop

\[
\text{for } i \leftarrow b, e \text{ do } \\
\quad \text{statement (or sequence)} \\
\text{end for}
\]

Equivalent to:

\[
i \leftarrow b \\
\text{while } i \leq e \text{ do } \\
\quad \text{statement (or sequence)} \\
\quad i \leftarrow i + 1 \\
\text{end while}
\]
For-each Loop

\[
\text{for each element of some collection } \textbf{do} \\
\quad \text{statement (or sequence)} \\
\text{end for}
\]

Equivalent to:

\[
i \leftarrow 1 \\
e \leftarrow \text{the number of elements in the collection} \\
\textbf{while } i \leq e \textbf{ do} \\
\quad \text{element } \leftarrow \text{ith element of the collection} \\
\quad \quad \text{statement (or sequence)} \\
\quad \quad i \leftarrow i + 1 \\
\textbf{end while}
\]
Example of a For-each Loop

1: Let $Grades$ be a sequence or list of grades
2: input $Grades$
3: for each $grade$ in $Grades$ do
4:   if $grade > 64$ then
5:     output pass
6:   else
7:     output fail
8: end if
9: end for
Algorithm for Binary-to-Decimal Conversion

1: Let $D$ be a decimal number, set to zero
2: Let $B$ be a binary number, set to zero
3: **input** $B$
4: Let $B'$ be $B$ with its digits reversed
5: $i \leftarrow 0$
6: **for each** binary digit $b \in B'$ **do**
7: \hspace{1cm} $D \leftarrow D + b \cdot 2^i$
8: \hspace{1cm} $i \leftarrow i + 1$
9: **end for**
10: **output** $D$
Program for B2D Conversion in Julia

D = 0
B = {1,1,0,0,1}
BPrime = B[end:-1:1]
i = 0
for b in BPrime
    D = D + b * 2^i
    i = i + 1
end
println( D )
Program for B2D Conversion in C

```c
#include <stdio.h>
#include <math.h>

int main()
{
    int b[] = { 1, 0, 0, 1, 1 }; 
    int n = 5; 
    int d = 0; 
    int i = 0; 
    for ( i = n - 1; i >= 0; i = i - 1 ) { 
        d = d + b[i] * (int) pow( 2.0, i ); 
    } 
    printf( "%d\n", d ); 
    return 0; 
}
```
Algorithm for Decimal-to-Binary Conversion

1: Let $B$ be an empty sequence of binary digits
2: Let $D$ be a decimal number, set to zero
3: input $D$
4: while $D \neq 0$ do
5: \hspace{1em} $r \leftarrow D \mod 2$
6: \hspace{1em} Add $r$ as the left-most digit of $B$
7: \hspace{1em} $D \leftarrow D \div 2$ (integer division)
8: end while
9: output $B$
Program for D2B Conversion in Julia

B = {}
D = 25
while D > 0
    r = D % 2
    unshift!( B, r )
    D = div( D, 2 )
end
println( B )
Program for D2B Conversion in C

```c
#include <stdio.h>
#include <math.h>

int main()
{
    int d = 25;
    int n = (int) ceil( log2( d ) );
    int b[n];
    int i = n - 1;
    int r = 0;
    while ( d > 0 ){
        r = d % 2;
        b[i] = r;
        i = i - 1;
        d = d / 2;
    }
    for ( i = 0; i < n; i = i + 1 ){
        printf( "%d", b[i] );
    }
    printf( "\n" );
    return 0;
}
```