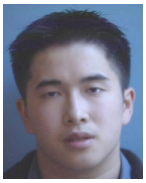
CS107
Introduction to Computer Science

Lecture 3, 4

An Introduction to Algorithms:
Loops

Administrativia

- **Study group**
  - Leader: Richard Hoang'05
  - Location: Searles 127/128
  - Time: Mondays 7-9pm



Building algorithms

- **Basic (primitive) instructions**
  - Read the input from user
  - Print the output to the user
  - Cary out basic arithmetical computations
- **Conditional instructions**
  - Execute an instruction (or a block of instructions) if a condition is true
- **Repeat (loop) instructions**
  - Execute a block of instructions (multiple) times until a certain condition is met
- and…**variables**

Basic instructions

1. Assign values to variables using basic arithmetic operations

- x = 3
- y = x/10
- z = x +25

2. Get input from user

- Get x

3. Print to user (screen)

- Print "The value of x is " x

Conditional instructions

- If (condition) then <instruction(s)>
  [else <instruction(s)>]

Examples:

```
if ((temp >= 80 and (temp < 100)) then print "Nice summer day"
else print "Chilly.."

if (a > b) then   if (a>c) then print "largest is" a
                  else print "largest is" c
else ...
```

Exercise

Write a program that asks the user for three numbers and prints out the largest. For example:

```
Enter first number: 10
Enter the second number: 25
Enter the third number: 5
The largest is 25
Goodbye.
```

Loop instructions

- A loop instruction specifies a group of statements that may be done several times (repeated):

  while <condition>

  < statements to be repeated >

- How does this work?
  - Condition is evaluated
  - If it is false than the loop terminates and the next instruction to be executed will be the instruction immediately following the loop
  - If it is true, then the algorithm executes the *<instructions to be repeated>* in order, one by one

Example

- What does this algorithm do?

```
Variables: count, square
set count = 1
while (count  <= 10)
        square = count *count
        print square, count
        count = count + 1
end
```

- Note: indentation
  - Used to delimit the block of instructions that are repeated
  - makes reading/understanding algorithms easier

Computing the sum 1+2+….+n

Write an algorithm which reads a positive integer from the user and computes the sum of all positive integers smaller than or equal to the number entered by the user.

Example: if the user enters 10, the algorithm should compute 1+2+3+…+10

Please enter a positive number: 10
The sum of all integers up to 10 is: 55
Goodbye.

Gauss formula

- We can actually find a formula for 1 + 2 + ……. + n

Gauss noticed that

- 1 + n = n+1
- 2 + (n-1) = n+1
- ….

==> 1 + 2 + … + (n-1) + n = n(n+1)/2

Computing the sum

One possible algorithm:

Variables: n(the number entered by the user), sum, i

- print "Please enter a positive integer"
- get n
- sum=0, i=0
- while (i <= n)
  - sum = sum + i
  - i = i+1
- Print "The sum up to " n " is: " sum

- An algorithm is not unique!!!
- There are many ways to solve a problem
- Moreover, given a certain way to solve a problem, there are many ways to express that into pseudocode!

- Style:
  - Give variables meaningful names
  - Write explanations/comments of what your algorithm does

Exercises

Given a number n from the user, write an algorithm..

- To compute the sum of all numbers strictly smaller than n
- To compute the sum of all even numbers <= n
- To compute the sum of all odd numbers <= n
- To compute the product of all numbers <= n (starting at 1)

Exercise

Write an algorithm that asks the user for a positive number. Assume the user is dumb (or stubborn) and enters a negative number. The program should keep asking the user for a positive number until the number entered by the user is positive. For example:

```
Enter a positive number: -3
Sorry, -3 is not positive.
Enter a positive number: -10
Sorry, -10 is not positive.
Enter a positive number: -2
Sorry, -2 is not positive.
Enter a positive number: 10
Finally. Goodbye.
```

Exercise

- Modify your previous algorithm so that the user keeps track of how many times the user enters a "wrong" number. Make it print this at the end.
- Now make it terminate if the user does not enter a "right" number within 10 attempts.

Exercise

Write an algorithm that asks the user for 10 temperature measurings, and prints out the temperatures entered and their average. For example:

```
Enter 10 temperatures: 21, 32, 34, 56, 67, 89, 21, 45, 67, 54
The recorded temperatures are: 21, 32, 34, 56, 67, 89, 21, 45,
67, 54. The average is: ...
```

Exercise

- Change your previous program so that it handles
  - 20 temperatures
  - 50 temperatures
  - 100 temperatures
  - 1000 temperatures

List variables

- How to represent inputs of arbitrary size?
- Suppose that we need to read 100 numbers from the user, or 1000, or..
  - we could give each variable a different name…tedious!!
- Use a **list variable**:
  - Variable: list a of size n
  - This means that a is a list of n elements: a_1, a_2, a_3,...,a_n
  - To read the list from the user use a loop to read each element
  - To print the list use use a loop to print each element
  - We can treat each element in the list as a variable
    - Set a_3 to 5
    - Set a_4 to a_3 +2
    - If (a_4 == a_5) then print "equal"

Lists

- Reading a list from the user

```
Print "enter size of list"
get n
i = 1
while (i <= n)
    print "enter element" i
    get a_i
    i = i+1
```

- Printing a list of size n to the user

```
i = 1
while (i <= n)
      print a_i
      i = i+1
```

Example

- What does the following code do?

```
n = 10
i = 1
while (i <= n)
          a_i = i
          i = i+1
x = 0
```

- What does the following code do?

```
i =1
while (i<= n)
        x = x + a_i
        i = i+1
print x
```

Example

- What does the following do?

```
n = 10
print "Enter " n " numbers:"
i = 1
while (i <= n)
          get a_i
          i = i+1
x = 0
i =1
while (i<= n)
        x = x + a_i
        i = i+1
print x
```

Searching

- Problem: Write an algorithm that reads from the user a list of 100 numbers and a target value, and searches to see if any element in the list is equal to the target value. If not, prints "target not found". If yes, prints "target found".

Searching, variations

- Modify your search algorithm so that:
  - It prints the location (index in the list) where it finds the target
  - It finds only the first occurence of target
  - It finds all occurences of target (and prints their locations)
  - It counts the number of occurences of target in the list
  - It counts how many elements in the list are larger than target

More exercises

- Write an algorithm that reads a list of 100 numbers from the user and
  - prints out the average of all numbers in the list.
  - prints out the largest element in the list
  - prints out the smallest element in the list