

CSE1301  
Computer Programming  
Lecture 12:  
Algorithm Design

1

Recall

- What is an algorithm?
- What is a procedure?
- What is top-down design?

2

Topics

- Functions
- Using functions in top-down design

3

Functions

- A named **sequence of instructions**
- Also known as: modules, procedures, subroutines, ...
- Give a name to a standard, frequently used sequence of actions
- Specify ("call") that sequence by name

4

Function Definition

**Define a function as:**

```
<functionname>  
{  
    <sequence>  
}
```

5

Example: Inviting Sam to a party

**Function Definition**

```
InviteToParty  
{  
    dial 9876-5432  
    say "Hello Sam, it's "  
    sayMyName()  
    say "Would you like to come to my party on 10 April?"  
    say "It's at 1 Wellington Road."  
    say "Great! See you then. Bye Sam"  
    hangUp()  
}
```

6

### Function Parameters

- Functions may have parameters
- They specify variations from call to call
- So that the same function can do different things
- Depending on the value of the parameters

7

### Function Parameters - continued

- For example:
  - $4 = 2$
  - $36 = 6$
- Both the above can be thought of as "calls" to the square root function
- But one returns 2, the other returns 6
- Depending on the value of the parameter

8

### Function Definition

**Define a function with parameters as:**

```
<functionname> ( <parameter>,  
  <parameter>, ... )  
{  
  <sequence>  
}
```

9

### Example: Inviting someone to a party

#### Function Definition

```
inviteToParty ( person, date, place)  
{  
  ringUp(person)  
  askToParty(date, place)  
  sayGoodbye(person, date)  
}
```

10

### Example: Inviting someone to a party

#### Function Definition

```
ringUp( person )  
{  
  set number to lookUpNumber(person)  
  dial(number)  
  say "hello,"  
  say person  
  say "it's"  
  sayMyName()  
}
```

11

### Example: Inviting someone to a party

#### Function Definition

```
askToParty(date, location)  
{  
  say "Would you like to come to my party on"  
  say date  
  say "It's at"  
  say location  
}
```

12

### Example: Inviting someone to a party

#### Function Definition

```
sayGoodbye ( person, date )  
{  
    say "Great! See you then. Bye"  
    say person  
    hangUp()  
}
```

13

### Top Down Design and Functions

- **Bottom Up Design**
  - Determine what simple sequences you will need to solve the problem
  - Code those sequences from primitives
  - Build more complex sequences using the simpler ones as "pseudo-primitives"
  - Continue building increasingly complex sequences
  - Stop when you have a sequence which solves the entire problem

14

### Top Down Design and Functions (cont)

- Build simple functions first
- Then use them as building blocks for more complex functions
- Example: Juggling

15

### Top Down Design and Functions (cont)

- Top-Down and Bottom-Up Design are really two sides of the same coin
- Example: Getting a Degree - Top-Down
  - You don't just walk in and pick up the piece of paper
  - You do first year, second year, third year (maybe fourth year), then hopefully you can get your degree!

16

### Top Down Design and Functions (cont)

- But how do you do first year?
  - First semester, second semester
- How do you do first semester?
  - 4 different subjects...
- How do you do a subject?
- ...

17

### Top Down Design and Functions (cont)

- Getting a degree - Bottom-Up
- Do an interesting-looking subject
- Do another...
- Keep doing them until you have enough for a degree...

18

### Top Down Design and Functions (cont)

- Obviously you need *direction*
- Bottom-up degree-getting strategies might never see you with the right combination of subjects to actually graduate!
- So elements of Top-down guidance are needed - what are we aiming at?

19

### Reading

- D & D, 5.1 – 5.2
- Brookshear, 6.4 (pp 244-245)

20