GCSE OCR

Computer Science
J277

5

Procedures and functions

Unit 7 Programming

PG ONLINE

Objectives

- Understand the concept of subroutines
- Learn how to write simple procedures and functions
- Understand and use parameters to pass data to procedures and functions
- Know that subroutines may use local variables which are accessible only within the subroutine
- Use local variables and explain why it is good practice to do so
- Explain the advantages of using subroutines in programs

Starter

- A computer program needs a menu to appear
  - They write the following code

```
print("menu")
print("1: New file")
print("2: Edit file")
print("3: Delete file")
print("4: Quit")
print("5: Show menu")

choice = input()
```

```
if choice == "5" then
    print("1: New file")
    print("2: Edit file")
    print("3: Delete file")
    print("4: Quit")
    print("5: Show menu")
endif
```

  - What is wrong with this code?
  - What alternative could be used?

Starter

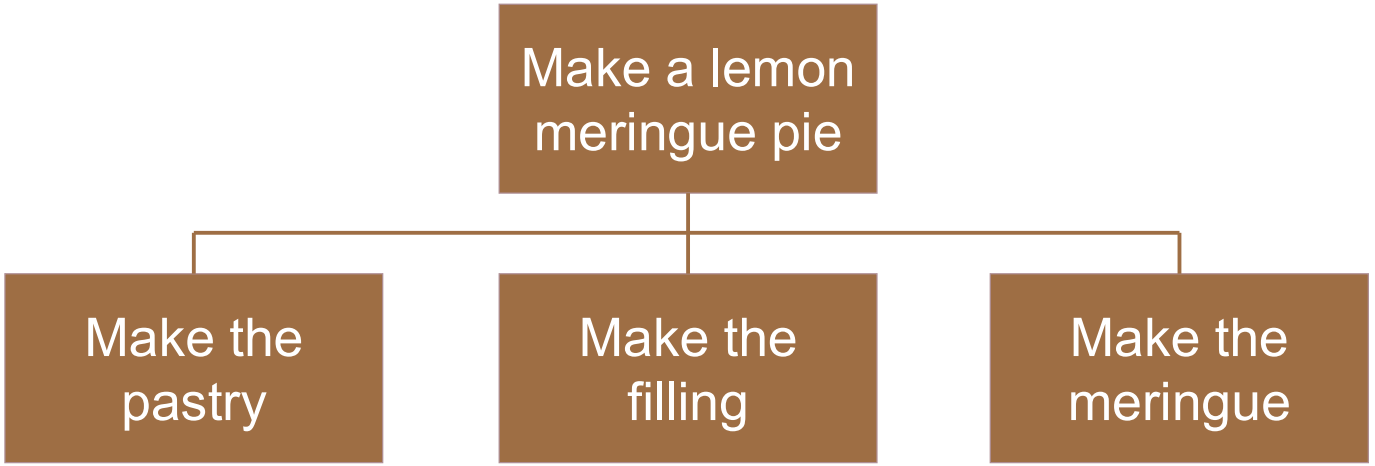
Answers

```
print("menu")
print("1: New file")
print("2: Edit file")
print("3: Delete file")
print("4: Quit")
print("5: Show menu")

choice = input()

if choice == "5" then
    print("1: New file")
    print("2: Edit file")
    print("3: Delete file")
    print("4: Quit")
    print("5: Show menu")
endif
```

- What is wrong with this code?
  It has copied and pasted parts of the code which can lead to mistakes being introduced
- What alternative could be used?
  Subroutines – in this case a procedure

Using subroutines

- As your programs become larger and more complex, they need to be broken down into smaller, self-contained sections
- Here is an illustration of the concept:



Functions and procedures

- Subroutines allow code that you intend to use a number of time to be grouped together under one name
  - Both functions and procedures are subroutines
  - Values can be passed to both procedures and functions
  - Functions will return a value after processing has taken place

Built-in subroutines

- All programming languages have built-in procedures and functions to perform common tasks
  - You have already used built-in subroutines

```
print("Hello")

input("Type in your name: ")


random(1,6)
```

- For each of the above subroutines, explain whether they are functions or procedures

Built-in subroutines

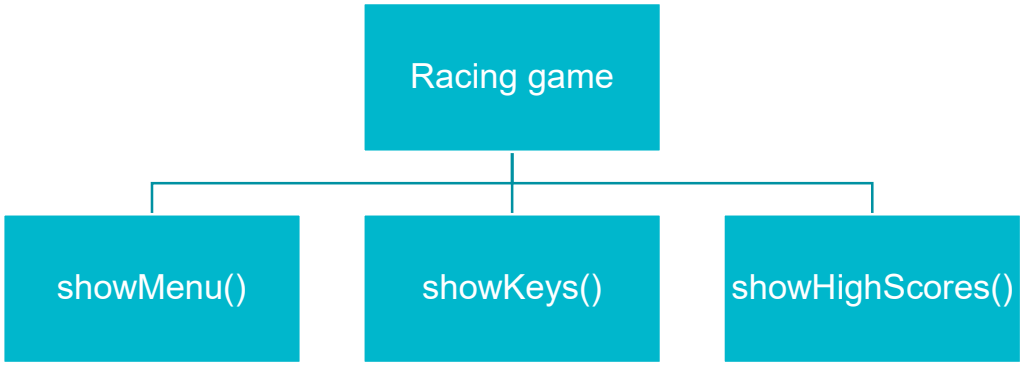
Answers

```
print("Hello")              //a procedure to output text

input("Type in your name: ")      //a function that returns
                                  //what the user enters

random(1,6)                 //a function that returns an
                            //integer between 1 and 6
```

- Procedures execute code in a subroutine – but they don't return anything
  - Functions will return something – so input returns what the user typed in, random returns a number
- Both functions and procedures can have arguments:
  - Arguments above are: "Hello", "Type in your name: ", 1, 6

A program using procedures

- A racing game requires the following:
  - Display a menu
  - Display the keyboard controls for the game
  - Display high scores in the game



showMenu procedure

```
procedure showMenu()
    print("      Menu ")
    print("==================")
    print("1: Play game")
    print("2: Show key controls")
    print("3: High scores")
    print("4: Return to main menu")
    print("5: Exit game")

showMenu()
```

This calls the procedure named showMenu

- How many parameters does showMenu have?
  - Write a procedure that will show the keyboard controls of the game

showKeys procedure

Answers

```
procedure showKeys()
    print("Keyboard controls")
    print("==================")
    print("1:   Up arrow - forwards")
    print("2: Down arrow – backwards/brake")
    print("3: Left arrow – turn left")
    print("4: Right arrow – turn right")
    print("5: R - refuel")

showKeys()
```

- Both showKeys and showMenu have no parameters

Worksheet 5

- Now complete **Task 1** on **Worksheet 5**

Functions

- Functions work just like procedures, except at the end they will return a value

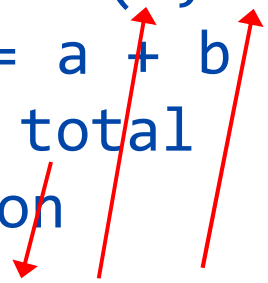
```
function sum(a, b)
   total = a + b
   return total
endfunction

answer = sum(5, 3)
print(answer)
```

- What is the output from the above function?
  - Explain how it works

Functions

Answers

```
function sum(a, b)
    total = a + b
    return total
endfunction
answer = sum(5, 3)
print(answer)
```



- First sum(5,3) calls the sum function with the arguments 5 and 3
- The sum function adds 5 + 3 and stores 8 in total
- The sum function returns the result of 8 to the function call
- This is then assigned to the variable called answer
- answer (which contains 8) is then output

- Output: 8
- Write a function named 'greatest' that has three parameters which are numbers
  - The function needs to return the greatest of the three numbers

Functions

Answers

```
function greatest(a, b, c)
    if a > b AND a > c then
        return a
    if b > a AND b > c then
        return b
    else
        return c
```

Passing parameters

- When you create a function with the statement:

```
sum(a, b)
```

- a and b are known as **parameters**

Try this problem:

An average score is calculated by dividing a total score by the number of attempts

- Create a function to calculate an average score

Pseudocode for average

Answers

```
function averageScore(totalScore, numScores)
 avg = totalScore / numScores
 return avg
endfunction
```

- To call the function:

```
total = input("Enter total score: ")
n = input("Enter number of scores: ")
print(averageScore(total, n))
```

Worksheet 5

- Now complete **Task 2** and **Task 3** on **Worksheet 5**

Why use subroutines?

- They are useful to break up a large program into self-contained units
  - Each subroutine can be tested separately to make sure it works correctly
  - Many programmers can work on a large program at the same time, cutting down development time
  - Subroutines can be re-used in other programs
- What is one other advantage of using subroutines?

More advantages of using subroutines

Answers

- Subroutines can be stored in a subroutine library and used in different programs if required
- Program maintenance is easier – if requirements change then just the affected subroutines need to be modified

Local variables and scope

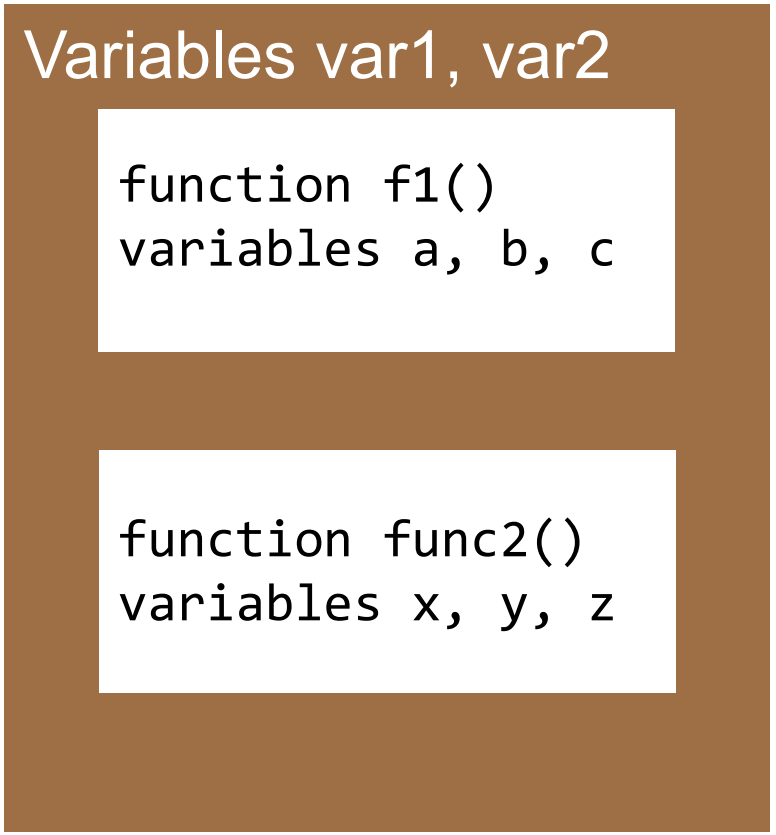
- The “scope” of a variable, constant, procedure or function defines the parts of the program in which it is recognised and can be used
- When you use an inbuilt function or procedure, you can write, for example

```
x = sqrt(y)
```

- In the random function you don’t need to worry about the internal variables or how they are used
  - This is because the variables used are only recognised within the function itself

Variable scope

- Variables var1 and var2 are global variables and can be seen anywhere in the program
- Variables a, b and c can only be seen and used inside f1
- Variables x, y and z can only be seen and used inside f2



Variable scope

- Think of each function as being a house with reflective windows – you can see out, but no one outside can see the variables in the “house”

Variable scope

Function A

Function B

Function C

- If you are in the main program, or in one of the functions, you cannot see any variables in the other functions
- If you are in a function “house”, you can see global variables in the main program, and you can use them but you can’t change them

Local and global variables

- Local variables only exist while the subroutine is executing
  - They are only accessible within the subroutine
- Global variables are accessible anywhere in the program
  - They are written in pseudocode using the global keyword:

    global score

What is printed by this program?

```
function changeNum()
    num = 6
    print(num)
endfunction
#Main program
num = 5
print(num)
#call function changenum
changeNum()
print(num)
```

This is a local variable

Prints 6

This doesn't have the global keyword so isn't affected by anything happening in changeNum

Prints 5

Calls changeNum

Prints 5 – the num variable in changeNum is different

Example

- Identify the local variables in function calcmax

```
function calcmax(heights)
    max = heights[0]
    n = heights.length
    for count = 0 to n – 1
       if heights[count] > max then
           max = heights[count]
       endif
    next count
    return max
endfunction
```

- What type of data structure is heights?
- How will the function be called?

Answer

Answers

- max, n and count are all local variables

```
function CalcMax(heights)    # heights is an array
    max = heights[0]
    n = len (heights)
    for count = 0 to n – 1
       if heights[count] > max then
            max = heights[count]
       endif
    next count
    return max
endfunction
```

- The function is called and the result stored with a statement such as:

  tallest = CalcMax (pupilHeights)

- The value max will be returned to tallest

Global and local constants

- Constants follow the same rules as variables
- A constant declared in the main program can be seen and used, but not changed, in any function

```
function addVAT(netPrice)
    vat = netPrice * VAT_RATE
    priceWithVAT = netPrice + vat
    return priceWithVAT
endfunction

global VAT_RATE = 0.20
priceNoVAT = 50.00
sellingPrice = addVAT(priceNoVAT)
print(sellingPrice)
```

Advantages of using local variables

- Local variables keep a subroutine self-contained, so it can be used in any program without variable names conflicting with those used in the calling program
  - Generally, you should use local variables wherever you possibly can

Worksheet 5

- Now complete **Task 4** on **Worksheet 5**

Plenary

- Answer the following questions in pairs
  - What are two types of subroutine used in programming?
  - How many parameters can functions and procedures have?
  - Explain the difference between a local and global variables
  - What are three reasons that you should try to use subroutines where possible?

Plenary

Answers

- Two types of subroutine used in programming:
  - Functions and procedures
- How many parameters can functions and procedures have? – Zero, one or many
- Explain the difference between a local and global variables
  - Globals are accessible anywhere in the program – locals only within the subroutine in which they were created
- What are three reasons that you should try to use subroutines where possible?
  - Reuse code, decomposition, more maintainable code

Copyright

Licence agreement