

SISTEME DISTRIBUITE

LABORAOTR 2 – Canale pentru conectări concurente

Săptămâna 18.10.2021 20.10.2021

Dr. Marius Iulian Mihailescu

marius-iulian.mihailescu@g.unibuc.ro

1. Să se implementeze o aplicație care să transfere cu success un mesaj (ex.: salut) de la o rutină *go (goroutine)* la alta prin intermediul unui canal de comunicare.
2. Să se creeze o aplicație care primește două mesaje pe același canal de comunicare.
3. Creați o aplicație care să notifice o altă rutină *go* că procesarea unei funcții a fost efectuată cu success.
4. Implementați o aplicație care să folosească un canal pentru primirea datelor și altul pentru trimiterea datelor.
5. Implementați o aplicație care să folosească instrucțiunea *select* cu scopul de a combina două rutine *go (goroutines)* pentru trimiterea de două sau mai multe mesaje.
6. Să se implementeze o aplicație în care presupunem că executăm un apel extern care turnează rezultatul său pe canalul 1 după 2 secunde.

Rezolvările se află pe paginile următoare.

Problema 1

```
package main

import "fmt"

func main() {

    messages := make(chan string)

    go func() { messages <- "salut" }()

    msg := <-messages
    fmt.Println(msg)
}
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: `main.go 1` and `channel1.go`.
- Code Editor:** Displays the Go code for `channel1.go`. The code creates a channel, starts a goroutine to send "salut" to it, and then reads from the channel.
- Terminal:** Shows the output of the command `go run "d:\GO Apps\channel1.go"`, which prints `salut`.
- Status Bar:** Shows the file is `master*`, the version is `Go 1.17.2`, there are 2 changes and 0 errors, and the current position is `Ln 9, Col 35`.

Problema 2

```
package main

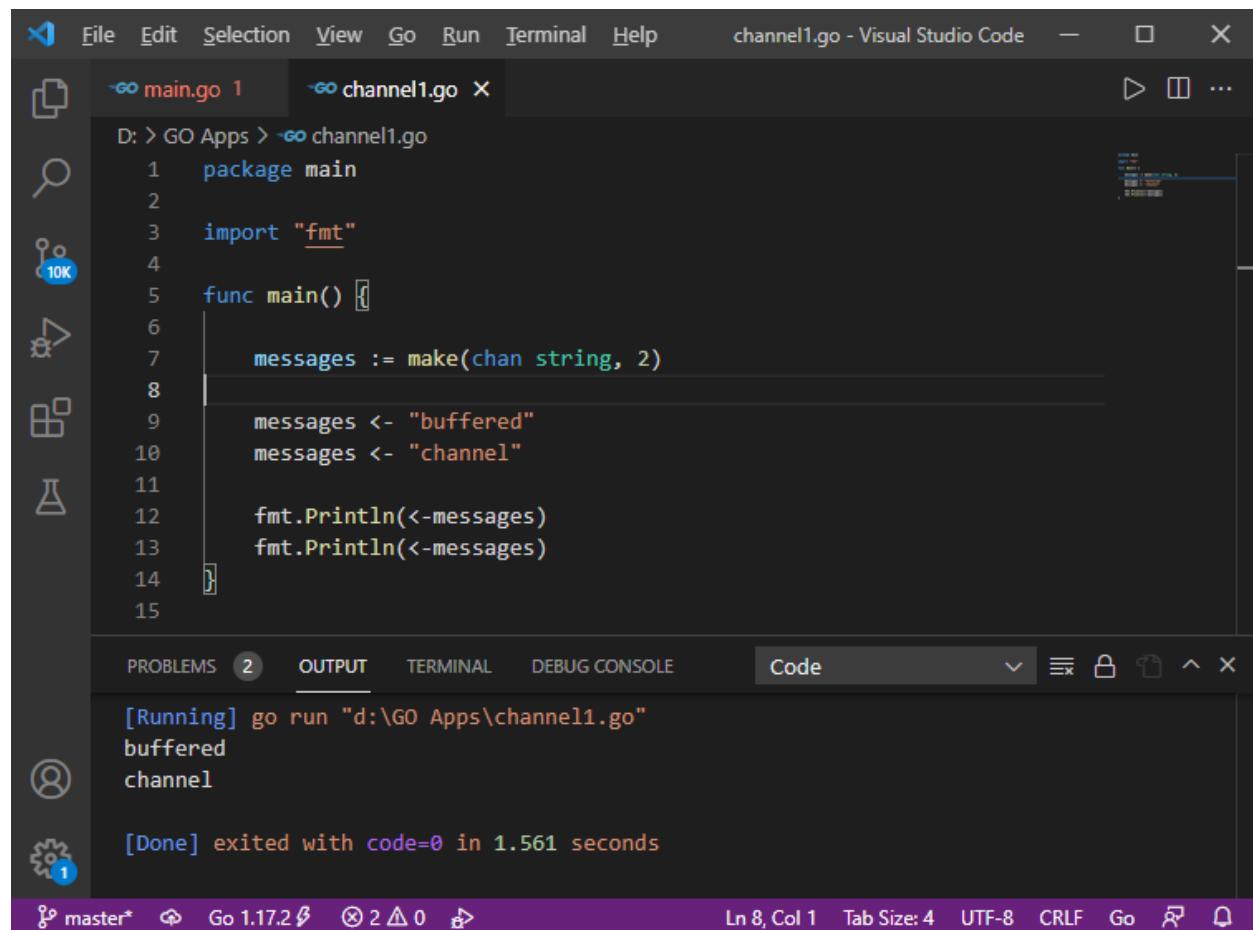
import "fmt"

func main() {

    messages := make(chan string, 2)

    messages <- "buffered"
    messages <- "channel"

    fmt.Println(<-messages)
    fmt.Println(<-messages)
}
```



The screenshot shows the Visual Studio Code interface with the Go extension installed. The left sidebar has icons for file, search, and code navigation. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar says "channel1.go - Visual Studio Code". The main editor area displays the Go code from the previous snippet. The bottom status bar shows "master*", "Go 1.17.2", "2 △ 0", "Ln 8, Col 1", "Tab Size: 4", "UTF-8", "CRLF", "Go", and a few other icons.

```
File Edit Selection View Go Run Terminal Help channel1.go - Visual Studio Code
D: > GO Apps > channel1.go
1 package main
2
3 import "fmt"
4
5 func main() {
6
7     messages := make(chan string, 2)
8
9     messages <- "buffered"
10    messages <- "channel"
11
12    fmt.Println(<-messages)
13    fmt.Println(<-messages)
14 }
15
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE Code [Running] go run "d:\GO Apps\channel1.go"
buffered
channel
[Done] exited with code=0 in 1.561 seconds

Ln 8, Col 1 Tab Size: 4 UTF-8 CRLF Go

Problema 3

```
package main

import (
    "fmt"
    "time"
)

func worker(done chan bool) {
    fmt.Println("working...")
    time.Sleep(time.Second)
    fmt.Println("done")

    done <- true
}

func main() {

    done := make(chan bool, 1)
    go worker(done)

    <-done
}
```

The screenshot shows the Visual Studio Code interface with two tabs open: `main.go` and `channel1.go`. The `main.go` tab contains the provided Go code. The `channel1.go` tab is visible in the background. The bottom right corner of the main window shows the status bar with the text: `[Running] go run "d:\GO Apps\channel1.go"`, `working...done`, and `[Done] exited with code=0 in 4.151 seconds`.

Problema 4

```
package main

import "fmt"

func primeste(pings chan<- string, msg string) {
    pings <- msg
}

func trimite(pings <-chan string, pongs chan<- string) {
    msg := <-pings
    pongs <- msg
}

func main() {
    pings := make(chan string, 1)
    pongs := make(chan string, 1)
    primeste(pings, "mesajul a fost trimis")
    trimite(pings, pongs)
    fmt.Println(<-pongs)
}
```

The screenshot shows the Visual Studio Code interface with two tabs open: `main.go` and `channel1.go`. The `main.go` tab contains the provided Go code. The `channel1.go` tab is currently active. In the bottom right corner of the editor, there is a status bar showing the output of the application's execution.

The output window displays the following text:

```
[Running] go run "d:\GO Apps\channel1.go"
mesajul a fost trimis

[Done] exited with code=0 in 1.35 seconds
```

The status bar at the bottom of the screen shows the current file is `master*`, the Go version is `Go 1.17.2`, and the build status is `2△0`. Other status bar items include `Ln 13, Col 1`, `Tab Size: 4`, `UTF-8`, `CRLF`, `Go`, and a few other icons.

Problema 5

```
package main

import (
    "fmt"
    "time"
)

func main() {

    canal1 := make(chan string)
    canal2 := make(chan string)

    go func() {
        time.Sleep(1 * time.Second)
        canal1 <- "primul mesaj"
    }()
    go func() {
        time.Sleep(2 * time.Second)
        canal2 <- "al doilea mesaj"
    }()
}

for i := 0; i < 2; i++ {
    select {
    case mesaj1 := <-canal1:
        fmt.Println("am primit", mesaj1)
    case mesaj2 := <-canal2:
        fmt.Println("am primit", mesaj2)
    }
}
}
```

The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar contains icons for file operations, search, and code navigation. The main editor window displays a Go program named `channel1.go`. The code uses channels to send and receive strings, with two goroutines producing messages and a select loop reading them. The output panel at the bottom shows the program's execution and its output: "am primit primul mesaj" and "am primit al doilea mesaj". The status bar at the bottom indicates the code is running in Go 1.17.2.

```
D: > GO Apps > -go channel1.go
1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 func main() {
9
10    canal1 := make(chan string)
11    canal2 := make(chan string)
12
13    go func() {
14        time.Sleep(1 * time.Second)
15        canal1 <- "primul mesaj"
16    }()
17    go func() {
18        time.Sleep(2 * time.Second)
19        canal2 <- "al doilea mesaj"
20    }()
21
22    for i := 0; i < 2; i++ {
23        select {
24            case mesaj1 := <-canal1:
25                fmt.Println("am primit", mesaj1)
26            case mesaj2 := <-canal2:
27                fmt.Println("am primit", mesaj2)
28        }
29    }
30 }
31
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE Code

```
[Running] go run "d:\GO Apps\channel1.go"
am primit primul mesaj
am primit al doilea mesaj

[Done] exited with code=0 in 3.988 seconds
```

master* Go 1.17.2 ⚡ 2 △ 0 ↻ Ln 12, Col 1 Tab Size: 4 UTF-8 CRLF Go ⚡ ⚡

Problema 6

```
package main

import (
    "fmt"
    "time"
)

func main() {

    canal1 := make(chan string, 1)
    go func() {
        time.Sleep(2 * time.Second)
        canal1 <- "rezultatul 1"
    }()

    select {
    case result := <-canal1:
        fmt.Println(result)
    case <-time.After(1 * time.Second):
        fmt.Println("timeout 1")
    }

    canal2 := make(chan string, 1)
    go func() {
        time.Sleep(2 * time.Second)
        canal2 <- "rezultatul 2"
    }()
    select {
    case res := <-canal2:
        fmt.Println(res)
    case <-time.After(3 * time.Second):
        fmt.Println("timeout 2")
    }
}
```

The screenshot shows a Visual Studio Code interface with a dark theme. The main area displays a Go file named `channel1.go`. The code uses channels to send and receive strings, with two select statements handling different types of results. The output panel at the bottom shows the application's execution and its results.

```
D: > GO Apps > -go channel1.go
4   import "fmt"
5   import "time"
6 )
7
8 func main() {
9
10    canal1 := make(chan string, 1)
11    go func() {
12        time.Sleep(2 * time.Second)
13        canal1 <- "rezultatul 1"
14    }()
15
16    select {}
17    case result := <-canal1:
18        fmt.Println(result)
19    case <-time.After(1 * time.Second):
20        fmt.Println("timeout 1")
21    }
22
23    canal2 := make(chan string, 1)
24    go func() {
25        time.Sleep(2 * time.Second)
26        canal2 <- "rezultatul 2"
27    }()
28    select {}
29    case res := <-canal2:
30        fmt.Println(res)
31    case <-time.After(3 * time.Second):
32        fmt.Println("timeout 2")
33    }
34
35
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE Code

```
[Running] go run "d:\GO Apps\channel1.go"
timeout 1
rezultatul 2

[Done] exited with code=0 in 6.371 seconds
```

master* Go 1.17.2 2 △ 0 Ln 16, Col 13 Tab Size: 4 UTF-8 CRLF Go ⚙ ⌂