Interaktiivsed programmid

Sõna äraarvamine — ”hangman”

```
Main> main
Enter a word: -------
k -------
h h------
m h---m--
o h---m--
n h-n-m-n
a han-man
g hangman
Enter a word:
```

Interaktiivsed programmid

- Tüüp `Interact`

```
type Interact = String -> String
```

- Funktsioon `interact` (eeldefineeritud prelüüdis)

```
interact  :: Interact -> IO ()
interact f = do input <- getContents
                putStr (f input)
```

- Näide:

```
import Char(toUpper)

capitalises :: Interact
capitalises = map toUpper

main = interact capitalises
```

"Hangman" (versioon 1)

```
hangman :: Interact
hangman input = "Enter a word: " ++ echo ++ "\n"
                ++ game word [] input'
        where echo   = ['-' | _ <- word]
              word   = before '\n' input
              input' = after  '\n' input

before x = takeWhile (/= x)
after  x = tail . dropWhile (/= x)

main = interact hangman
```

"Hangman" (versioon 1)

```
game :: String -> String -> Interact
game word guess (c:input) = line ++ rest
     where line   = [c] ++ " " ++ reveal ++ "\n"
           reveal = [dash w | w <- word]
           dash w | w `elem` (c:guess) = w
                  | otherwise          = '-'
           rest   | '-' `elem` reveal
                                = game word (c:guess) input
                  | otherwise = hangman input
```

Interaktiivsed kombinaatorid

```
readChar, peekChar :: (Char -> Interact) -> Interact
readChar prog (c:cs)   = prog c cs
peekChar prog cs@(c:_) = prog c cs

unreadChar :: Char -> Interact -> Interact
unreadChar c prog cs = prog (c:cs)

writeChar :: Char -> Interact -> Interact
writeChar c prog cs = c : prog cs

writeStr :: String -> Interact -> Interact
writeStr s prog cs = s ++ prog cs
```

Interaktiivsed kombinaatorid

```
readLine :: String -> (String -> Interact) -> Interact
readLine prompt g cs = prompt ++ g line input'
        where line   = takeWhile (/= '\n') cs
              input' = tail (dropWhile (/= '\n') cs)

end   :: Interact
end cs = ""

pressAnyKey :: Interact -> Interact
pressAnyKey prog = readChar (\c -> prog)

ringBell :: Interact -> Interact
ringBell  = writeChar '\BEL'
```

"Hangman" (versioon 2)

```
hangman = readLine "Enter a word: " $ \word ->
          let echo = ['-' | _ <- word] in
          writeStr (echo ++ "\n")   $
          game word []

game word guess = readChar $ \c ->
                  let dash w | w `elem` (c:guess) = w
                             | otherwise          = '-'
                      reveal = [dash w | w <- word]
                  in writeStr([c] ++ " " ++ reveal ++ "\n") $
                  if '-' `elem` reveal
                  then game word (c:guess)
                  else hangman
```

ANSI-ekraani kontrollimine

```
cls :: String
cls  = "\ESC[2J"

highlight  :: String -> String
highlight s = "\ESC[7m"++s++"\ESC[0m"

goto :: Int -> Int -> String
goto x y = "\ESC[" ++ show y ++ ";" ++ show x ++ "H"

home :: String
home  = goto 1 1
```

ANSI kombinaatorid

```
clearScreen :: Interact -> Interact
clearScreen  = writeStr cls

moveTo :: Pos -> Interact -> Interact
moveTo (x,y) = writeStr (goto x y)

writeAt :: Pos -> String -> Interact -> Interact
writeAt pos s = writeStr (at pos s)

readAt :: Pos -> String -> (String->Interact) -> Interact
readAt pos prompt g = moveTo pos $ readLine prompt g
```

"Hangman" (versioon 3)

```
hangman = clearScreen $
          readAt (1,1) "Enter a word: " $ \w ->
          let echo = ['-' | _ <- w] in
          writeStr (echo ++ "\n")   $
          game w []

newgame = writeStr "Start a new game? (y/n)" $
          readChar $ \c ->
          if toUpper(c) == 'Y'
          then hangman
          else end
```