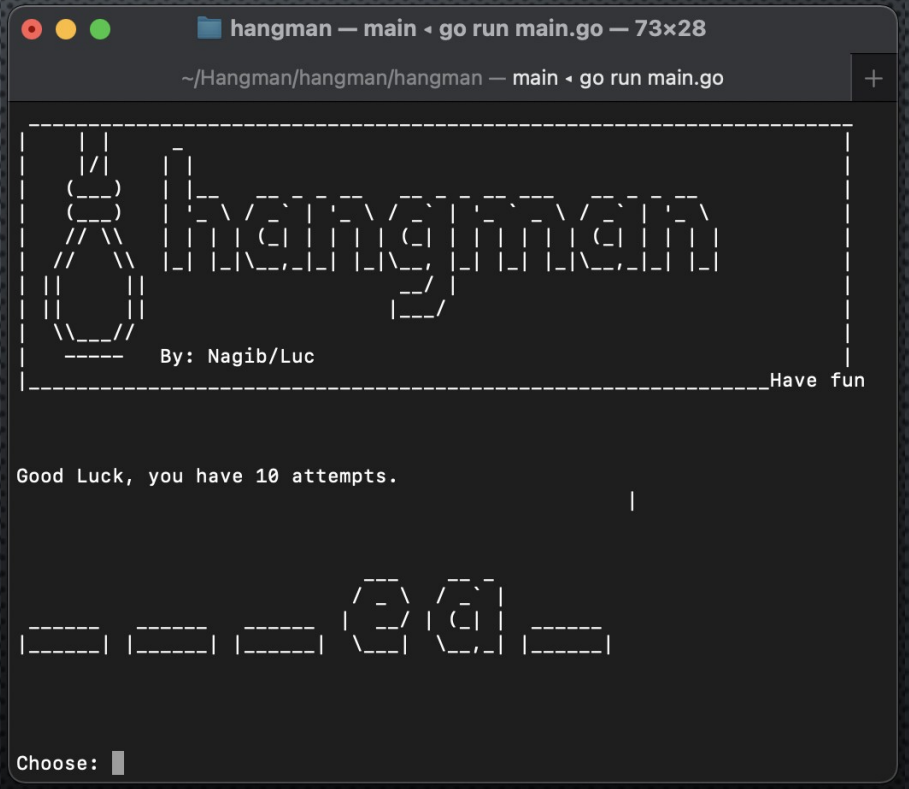
Hangman



Par Nagib & Luc

SOMMAIRE

- OBJECTIF : CRÉE UN HANGMAN EN LANGAGE GOLANG. 5 EXERCICES NOUS ON AIDÉ À ARRIVER À UNE VERSION FINAL COMPLÈTE.
- 1) HANGMAN CLASSIC – CRÉE UN JEU DANS LE TERMINAL – LE PENDUE
- 2) START & STOP – CRÉE UNE SAUVEGARDE DE LA PARTIE EN COURT
- 3) ADVANCED FEATURE – OPTIONS SUPPLÉMENTAIRES (…)
- 4) ASCII ART – PRINT LA SOLUTION SOUS LETTRES ASCII
- 5) HANGMAN DEFINITIVE - JUXTAPOSITION DE TOUT LES EXERCICES.
-
- RÉPARTITION DES TACHES COMPLÉMENTAIRE : TOUT À ÉTÉ FAIT À DEUX.

N

STRUCTURE DE DONNÉES

```
type hangm struct {
    Solution    []string
    Word        []string
    Chance      int
    Drawingpos int
}
```

- **SOLUTION**: SOLUTION MOT RÉCUPÉRÉ
- **WORD**: MOT INCOMPLET
- **CHANCE**: NOMBRE DE CHANCES RESTANTE
- **DRAWINGPOS**: NOMBRE RENVOYANT À L'EMPLACEMENT DU DESSIN (HANGMAN)

Début du programme:

Affichage de départ

Ouvre un fichier comportant les mots

Prend un mot au hasard

Incrémentation dans deux variables. Le mot complet, ainsi que le mot à rechercher « H_ _ L _ »

Renvoie à la fonction de lancement principale du jeu

```go
func main() {
    //_____________________________________
    aff, _ := os.Open("affichage.txt")
    scan := bufio.NewScanner(aff)
    scan.Split(bufio.ScanLines)

    for scan.Scan() {
        fmt.Println(scan.Text())
    }
    fmt.Println("")
    //_____________________________________
    args := os.Args[1:]
    var text []string
    min := 0
    max := 0
    rand.Seed(time.Now().UnixNano())
    file, err := os.Open("words.txt")
    scanner := bufio.NewScanner(file)
    scanner.Split(bufio.ScanLines)
    for scanner.Scan() {
        text = append(text, scanner.Text())
        max++ //borne max random
    }
    random := rand.Intn(max-min) + min
    word := text[random]
    max = len(word)
    if err != nil {
        log.Fatalf("failed to open")
    }
    tmpword, tmpsolution := (printn(word, min, max))
    hgn := hangm{tmpsolution, tmpword, 10, 0}

    if len(args) > 1 { //Récupération de la sauvegarde si flag trouvé
        if args[0] == "--startWith" && args[1] == "save.txt" {
            save, _ := ioutil.ReadFile("save.json")
            err = json.Unmarshal([]byte(save), &hgn)
            if err != nil {
                log.Fatalf("failed to encode game")
            }
        }
    }
    hangman(hgn)
}
```

N

FORMATAGE DES MOTS INITIALISATION DE 2 TABLEAUX

Initialisation du mot complet

Révèle n lettre du mot complet et remplace les emplacements vide par « _ »

```go
//initialisation du mot incomplet et de sa solution
func printn(word string, mi int, ma int) ([]string, []string) {
    wordarr := make([]string, len(word))
    solution := []string{}
    for _, v := range word { //met le mot complet dans solution
        solution = append(solution, string(v))
    }
    for i := 0; i < len(word)/2-1; i++ { //Révèle n lettres random du mot où n est len(word) / 2 - 1
        r := rand.Intn(ma-mi-1) + mi
        if wordarr[r] == "" {
            wordarr[r] = solution[r]
        }
    }
    for i := 0; i < len(wordarr); i++ { //Remplace chaque case vide par un "_"
        if wordarr[i] == "" {
            wordarr[i] = "_"
        }
    }
    return wordarr, solution
}
```

L

BOUCLE DU JEU 1/3

Implentation de standard.txt dans un slice

Permet de vérifier le cas de victoire (advanced features)

Phrase du début ou bien après la save

Affichage au départ du mot sous forme Ascii

Boucle tournant selon le nombre de chance restantes.
Vérification du mot si chaque lettre est bonne.
Alors le jeu est gagné.

```go
func hangman(hgn hangm) {
    textascii := []string{}
    file, err := os.Open("standard.txt")
    if err != nil {
        log.Fatalf("failed to open standard.txt")
    }
    scanner := bufio.NewScanner(file)
    scanner.Split(bufio.ScanLines)
    for scanner.Scan() {
        textascii = append(textascii, scanner.Text()) // Slice ascii
    }
    strsolution := ""
    for _, v := range hgn.Solution { //Incrémentation de la solutin
        strsolution += string(v)
    }
    inputletter := []string{}
    if hgn.Chance == 10 {
        fmt.Println("Good Luck, you have", hgn.Chance, "attempts.")
    } else {
        fmt.Println("Welcome Back, you have", hgn.Chance, "attempts remaining.")
    }
    asciiletter(textascii, hgn)
    for hgn.Chance > 0 {
        equal := true
        for i := 0; i < len(hgn.Word); i++ { //vérifie si le mot est complet
            if hgn.Word[i] != hgn.Solution[i] {
                equal = false
                break
            }
        }
        if equal == true { //si le mot est complet victoire et arrêt
            fmt.Println("Congrats !")
            os.Remove("resultat.txt")
            return
        }
        /*
            for _, x := range hgn.word { //print mot incomplet
                fmt.Print(string(x), " ")
            }
        */
```

N

BOUCLE DU JEU 2/3

Ici on récupère l'input du terminal

Si reçoit « STOP », arrêt de la partie et sauvegarde dans la fonction stopsave

Traitement si l'input est égal au mot. Sinon chance -2 et le dessin avance de 2

Traitement de l'input (lettre) et vérifie la lettre précédente sinon (err)

Permet de vérifier les lettres précédentes

```
fmt.Println("")
fmt.Println("")
fmt.Print("Choose: ")
scanner := bufio.NewScanner(os.Stdin)
scanner.Scan()
input := scanner.Text() //prend en entrée un mot ou une lettre
here := false

if stopsave(input, hgn) == "STOP" {
    break
}

if len(input) > 1 && input != "STOP" { //Traite input == mot
    if input == strsolution {
        fmt.Println("Congrats !")
        os.Remove("resultat.txt")
        return
    } else {
        hgn.Chance -= 2
        if hgn.Chance == 8 {
            hgn.Drawingpos += 8
        } else {
            hgn.Drawingpos += 16
        }
        fmt.Println("Not present in the word, ", hgn.Chance, " attempts remaining")
    }
} else { //Traite input == lettre
    for _, v := range inputletter {
        if v == input {
            fmt.Println("Error letter already submitted")
        } else {
            for i, v := range hgn.Solution {
                if input == v {
                    hgn.Word[i] = string(v)
                    here = true
                }
            }
        }
    }
    inputletter = append(inputletter, input)
}
```

BOUCLE DU JEU 3/3

Si l'input ne correspond pas à l'une dès lettres du mot chance -1. Et on passe au dessin suivant (+8 index)

Envoie à la func Ascii letter

Envoie à la func Dessin

Si plus de chances fin de la partie et supprime un fichier temporraire

```
    if here == false && len(input) <= 1 { //pas présent print chance-1 et dessin correspondant
        hgn.Chance--
        fmt.Println("Not present in the word, ", hgn.Chance, " attempts remaining")
        if hgn.Chance != 9 {
            hgn.Drawingpos += 8
        }
    }
    asciiletter(textascii, hgn)
    dessin(hgn)
}
if hgn.Chance == 0 {
    fmt.Println("You lost, try again")
}
os.Remove("resultat.txt")
}
```

Dessin

Va afficher le dessin selon ce qui est pris en compte dans la structure « hgn.Drawingpos ». Soit l'index du dessin définit précédemment.

Va lire le fichier où est le dessin

```
//print dessin correspondant
func dessin(hgn hangm) {
    var text []string
    file, err := os.Open("hangman.txt")
    if err != nil {
        log.Fatalf("failed to open")
    }
    scanner := bufio.NewScanner(file) //lis le fichier qui contient les dessins
    scanner.Split(bufio.ScanLines)
    for scanner.Scan() {
        text = append(text, scanner.Text())
    }
    for _, x := range text[hgn.Drawingpos : hgn.Drawingpos+8] { //Print le dessin ,via la dernière position drawingpos(l'index) défini dans la struct +8
        fmt.Println(string(x))
    }
}
```

ASCII ART 1/2

Création d'un .txt temporaire permettant de visualiser le résultat final et de mieux géré le print des index.

Création d'un SLICE alphabet permettant de définir l'emplacement de l'input dans l'alphabet. Afin de le comparé à l'emplacement de la lettre dans le fichier contenant les lettres Ascii.

Une boucle qui se référera au mot incomplet afin de le print en Ascii

Une deuxième boucle qui définira ainsi l'emplacement de la lettre et ira la chercher dans le fichier Ascii reçus dans l'entête de la fonction.

Si la lettre n'est pas trouver alors il print par défaut l'index de « _ » dans le fichier Ascii.
Puis remet à zéro le compteur.

```go
func asciiletter(ascii []string, hgn hangm) {
    file, err := os.OpenFile("resultat.txt", os.O_CREATE|os.O_WRONLY, 0600)
    defer file.Close() // on ferme automatiquement à la fin de notre programme //L'obj
    slicealpha := []string{}
    alpha := "abcdefghijklmnopqrstuvwxyz"
    count := 0
    essaie := []string{} //Ici min et max correspondent aux index séparant les lettre
    for _, x := range alpha {
        slicealpha = append(slicealpha, string(x))
    }
    for i := range hgn.Word {
        /*Si la lettre recu correspond à une dès lettre de lalphabet contenue dans le
alors (count) calcul la position de la lettre en question*/
        for j := range slicealpha {
            count++

            if hgn.Word[i] == slicealpha[j] {
                min := (count * 9) + 577
                max := (count * 9) + 586

                for _, x := range ascii[min:max] {
                    _, err = file.WriteString("\n")
                    _, err = file.WriteString(string(x))
                    // écrire dans le fichier "resultat.txt" la lettre correspondante
                    if err != nil {
                        panic(err)
                    }
                }
            }
        }
        if hgn.Word[i] != hgn.Solution[i] {
            for _, x := range ascii[116:125] {
                _, err = file.WriteString("\n")
                _, err = file.WriteString(string(x))
                // écrire dans le fichier "resultat.txt" le caractère "_"
                if err != nil {
                    panic(err)
                }
            }

        }
        count = 0
    }
```

ASCII ART 2/2

Cette boucle aura pour maximum 9 car chacune dès lettre est séparé en 9 index. Puis par la suite l'objectif étant d'aller chercher dans resultat.txt chaque Index n+1 de chacune des lettres.

Afin de pouvoir print à la suite chaque index de toute les lettres qui sont égal au mot incomplet « H _ _ L O ».

Nous nous sommes inspirer d'une imprimante afin de pouvoir réussir à print les lettres dans le bonne ordres.

(Soit essaie[n])

```
counter := 9
counter2 := 1
for j := 0; j < 9; j++ { //Recupère l'index+1 \n de cha
    for i := len(hgn.Solution); i > 0; i-- {
        counter2++
        n := 0
        n = 9*counter2 - counter
        filer, _ := os.Open("resultat.txt")
        scanner := bufio.NewScanner(filer)
        scanner.Split(bufio.ScanLines)

        for scanner.Scan() {
            essaie = append(essaie, scanner.Text())
        }
        fmt.Print(essaie[n])
    }
    if j != 9 {
        fmt.Print("\n")
    }
    counter2 = 0
    counter--
}
```

Stop & Save (fonction main):

Ici la fonction à pour but de sauvegarder la structure dans son état et de mettre un terme au programme.
(elle est appelé depuis la fonction qui gère l'input)

Si le flag est reconnu et réinitialise la structure via un fichier Json qui la contient

```
if len(args) > 1 { //Récupération de la sauvegarde si flag trouvé
    if args[0] == "--startWith" && args[1] == "save.txt" {
        save, _ := ioutil.ReadFile("save.json")
        err = json.Unmarshal([]byte(save), &hgn)
        if err != nil {
            log.Fatalf("failed to encode game")
        }
    }
}
hangman(hgn)
```

```
func stopsave(input string, hgn hangm) string {
    if input == "STOP" { //START AND STOP
        fmt.Println(hgn)
        save, _ := json.Marshal(hgn)
        f, _ := os.Create("save.json")
        f.Write(save)
        fmt.Println("Game Saved in save.json.")
        return "STOP"
    }
    return ""
}
```

DIFFICULTÉES/ AMÉLIORATION

- DIFFICULTÉES AVEC:

RÉADAPTER LE PROGRAMME EN STRUCTURE

- AMÉLIORATION POSSIBLES:

CRÉATION D’UNE API

N&L