Atelier de découverte du langage C

A – Jeu du « More Or Less »

L'objectif est de créer un jeu en langage C ou le joueur doit trouver un nombre aléatoire entre 0 et 100 en un minimum de coup.

Étape N°1 : catch de la saisie
Ouvrir un terminal
Lancer nano
Taper ces premières lignes pour capter une chaîne de caractères saisie par le joueur sur le clavier.

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>
#include <time.h>

int limit;

int get_input(void)
{
    char *str = NULL;
    size_t size = 0;
    int red;
    int result;

    /* Get input */
    red = getline(&str, &size, stdin);
    printf("Before :%s\n", str);
    str[strlen(str) - 1] = 0; // Delete new line
    //printf("After :%s\n", str);

   return (result);
}

int main(void)
{
   get_input() ;
}
```

Enregistrer ce code dans un fichier : moreorless.c
Dans le terminal, compiler ce premier code avec la commande : gcc moreorless.c
Lancer le fichier généré : ./a.out
Tester en saisissant des chiffres ou des lettres.

Étape N°2 : Contrôle de la saisie
Nous allons rajouter une fonction qui va tester que la saisie est bien un nombre

Dans un terminal, ouvrir le fichier moreorless.c avec nano
Saisir le fonction de test : is_str_num...

```
int limit;
bool is_str_num(char *str)
{
    for (int i = 0; str[i]; i++)
        if (str[i] < '0' || str[i] > '9')
            return (false);
    return (true);
}
int get_input(void)
...
```

et faite appel à cette fonction dans la fonction get_input :

```
    printf("After :%s\n", str) ;

   /* Verif is number */
    if (!is_str_num(str)) {
        puts("Error: Input should be a unsigned int");
        return (-1);
    }

    /* Get input to integer */
    result = atoi(str);
   return (result);
...
```

Enregistrer ce code dans un fichier : moreorless.c
Dans le terminal, compiler ce premier code avec la commande : gcc moreorless.c
Lancer le fichier généré : ./a.out
Tester en saisissant des chiffres ou des lettres.

Étape N°3 : Contrôle de la saisie
Nous allons maintenant tester que la valeur saisie se trouve dans un intervalle limité à 100.

Dans un terminal, ouvrir le fichier moreorless.c avec nano
Compléter la fonction get_input avec le test suivant :

```
  /* Verif value to limite */
    limit = 100;
    if (result > limit) {
        printf("The number received from the input: %d is too higher, \should be less than the limit: %d\n",
result, limit);
        return (-1);
    }
 return (result);
...
```

Enregistrer ce code dans un fichier : moreorless.c
Dans le terminal, compiler ce premier code avec la commande : gcc moreorless.c
Lancer le fichier généré : ./a.out
Tester en saisissant des chiffres supérieurs à la valeur limite.

Étape N°4 : Définir un nombre aléatoire et tester en boucle la saisie par rapport à ce nombre
Dans cette étape nous allons définir un nombre aléatoire

Dans un terminal, ouvrir le fichier moreorless.c avec nano
Créer cette nouvelle fonction à la suite de la fonction get_input(void)

```
int get_random_nb(void)
{
    /* Define random list */
    srand(time(NULL));

    /* Get the random */
    return (rand() % limit);
}
```

Nous allons maintenant créer une nouvelle fonction qui va boucler la saisie en appelant la fonction get_input et inclure le test de comparaison avec la variable aléatoire.

```
void loop(int secret_nb)
{
    int input;

    while (true) {
        input = get_input();
        if (input != -1) {
            if (input > secret_nb)
                printf("The secret number is higher than %d\n", input);
            else if (input < secret_nb)
                printf("The secret number is lower than %d\n", input);
            else {
                printf("Congratulate\nYou found the secret number: %d\n", input);
                exit(0);
            }
        }
    }
}
```

La fonction main(void) doit être adaptée pour générer le nombre aléatoire et lancer ensuite la boucle de test :

```
int main(void)
{
    int secret_nb = get_random_nb();

    loop(secret_nb);
}
```

Vous pouvez à nouveau compiler le programme C et le lancer dans le terminal.

Étape N°5 : Améliorer le programme en créant par exemple un compteur de nombre d’essais

A vous de jouer:)

B – Jeu du Pendu

L'objectif est de créer un second jeu en langage C : le jeu du pendu ou le joueur doit trouver toute les lettres d'un mots en moins de 5 erreurs.

Étape N°1 : catch de la saisie
Comme sur le précédent programme nous allons commencer par créer la fonction de saisie et le main :

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>

char *get_input(void)
{
    char *str = NULL;
    size_t size = 0;
    int red;

    /* Get input */
    red = getline(&str, &size, stdin);
    str[strlen(str) - 1] = 0; // Delete new line
    return (str);
}


int main(void)
{
   char *lettre;
   lettre=get_input();
   printf (lettre);
}
```

Enregistrer ce code dans un fichier : pendu.c
Dans le terminal, compiler ce premier code avec la commande : gcc pendu.c
Lancer le fichier généré : ./a.out
Tester en saisissant des chiffres ou des lettres.

Étape N°2 : Le nom a trouver en paramètre de lancement

Nous allons apporter une modification significative au programme lui permettant de prendre un mots en entrée lors de son lancement.

Le main est alors modifié de la façon suivante :

```
int main(int ac, char **av)
{
    if (ac != 2) {
        puts("Error: \033[1;31mArgument needed");
        return (84);
    }
   char *lettre;
   lettre=get_input();
   printf (lettre);
}
```

Enregistrer ce code dans un fichier : pendu.c
Dans le terminal, compiler ce premier code avec la commande : gcc pendu.c
Lancer le fichier généré : ./a.out coucou
Nous avons dans le lancement inclus un paramètre qu'il est possible de modifier a chaque lancement.

Étape N°3 : Comparaison de la saisie et du mot en paramètre

Nous allons d'abord créer une fonction qui permet de comparer les lettres d'un mot.

```
bool is_char_found(char c, char *to_find)
{
    for (int i = 0; to_find[i]; i++)
        if (to_find[i] == c)
            return (true);
    return (false);
}
```

Puis nous allons créer une fonction de mise à jour des lettres trouvées du mot

```
void update_string(char c, char *to_find, char *line)
{
    for (int i = 0; to_find[i]; i++)
        if (to_find[i] == c)
            line[i] = to_find[i];
}
```

Puis nous allons créer la fonction loop qui va appeler à chaque saisie cette fonction de comparaison des lettre et celle de mise à jour :

```
void loop(char *to_find)
{
    int size = strlen(to_find);
    char line[size + 1];
    char *input;
    int nb_life = 5;

    /* Fill searching line */
    for (int i = 0; i < size; i++)
        line[i] = '_';
    line[size] = 0;

    /* True loop */
    while (true) {
        input = get_input();
        /* Verif if only one char */
        if (!is_char_found(input[0], to_find))
                nb_life--;
            else
                update_string(input[0], to_find, line);
            /* Display current result */
            printf("%s\nNb life: \033[1;34m%d\033[0m\n", line, nb_life);
        }
    }
}
```

il faut également modifier le main pour appeler la fonction loop :

```
int main(int ac, char **av)
{
    if (ac != 2) {
        puts("Error: \033[1;31mArgument needed");
        return (84);
    }
    loop(av[1]);
}
```

Enregistrer ce code dans un fichier : pendu.c
Dans le terminal, compiler ce premier code avec la commande : gcc pendu.c
Lancer le fichier généré : ./a.out coucou
Cela marche mais il n'y a pas de test de fin.

Étape N°4 : Les tests de Win et Lost

Nous allons compléter le programme avec une fonction de test de victoire :

```
bool is_win(char *line)
{
    for (int i = 0; line[i]; i++)
        if (line[i] == '_')
            return (false);
    return (true);
}
```

Cette fonction sera appelé dans la fonction Loop :

```
void loop(char *to_find)
{
    int size = strlen(to_find);
    char line[size + 1];
    char *input;
    int nb_life = 5;

    /* Fill searching line */
    for (int i = 0; i < size; i++)
        line[i] = '_';
    line[size] = 0;

    /* True loop */
    while (true) {
        input = get_input();
        /* Verif if only one char */
        if (!is_char_found(input[0], to_find))
                nb_life--;
            else
                update_string(input[0], to_find, line);
            /* Display current result */
            printf("%s\nNb life: \033[1;34m%d\033[0m\n", line, nb_life);

            if (nb_life == 0) {
            puts("\033[1;31mYou loose");
            exit(0);
        }
        if (is_win(line)) {
            puts("\033[1;32mYou win");
            exit(0);
        }
    }
}
```

Enregistrer ce code dans un fichier : pendu.c
Dans le terminal, compiler ce premier code avec la commande : gcc pendu.c
Lancer le fichier généré : ./a.out coucou

Étape N°5 : Améliorations
Le programme fonctionne mais il peut encore être améliorer.
Par exemple il est possible de saisir plusieurs lettres lors de la saisie. Il faudrait alerter le joueur lorsque le programme détecte plusieurs lettre saisie en même temps.
A vous de jouer !