HANGMAN

Description of the game :

Hangman is a problem which consists in guessing a word .

Before starting the game the user has to choose the level of the difficulty : he has to choose the number of letters in the word to guess and the allowed number of misses .
In the beginning a word is represented in the screen by a row of dashes, representing each letter of the word . The user is asked every time to write a letter and with every right guess the letter will be printed instead of the dash in its right position. If the letter occurs more than once in the word, then all dashes covering this letter will be unveiled. The user loses if the number of the wrongly guessed letters surpassed the allowed number of misses.
In each try, the word must be selected randomly from the file that is attached to the problem and should satisfy the level of difficulty chosen by the user.

At the end of each game the results will be saved in a file called history .

The functions which I will use in my program are :

Randomword

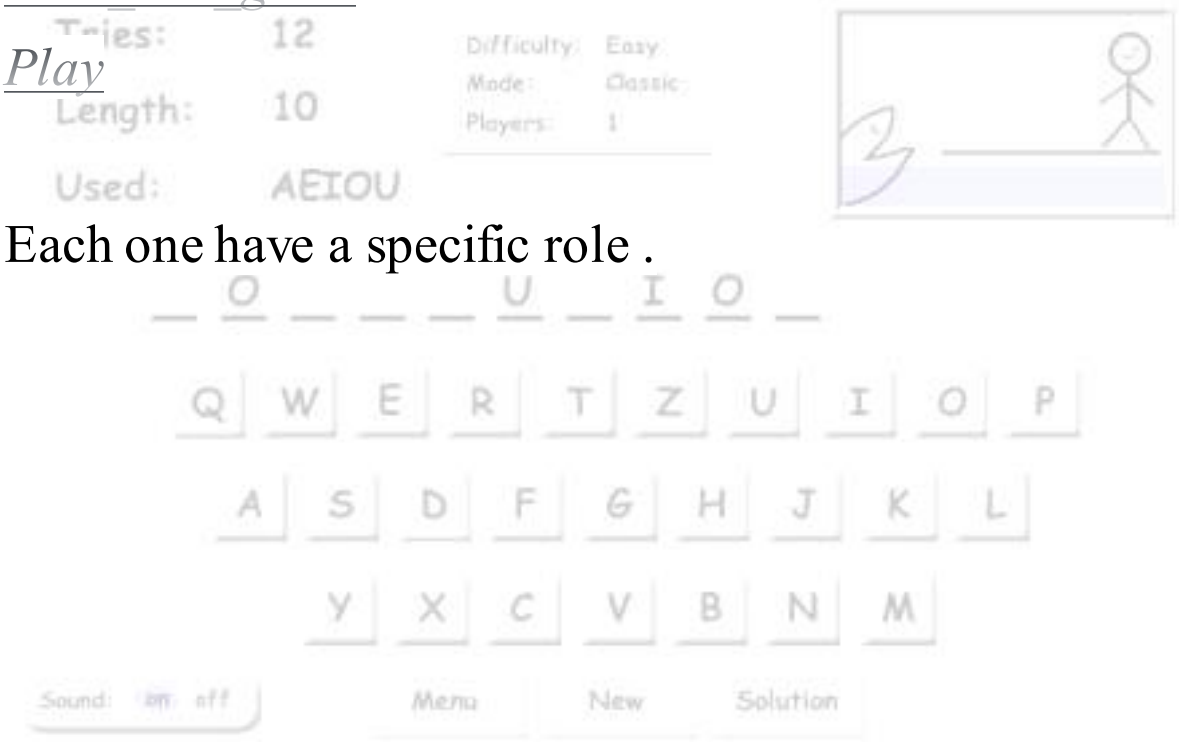
Notify

Modif

Show_result_and_time

Save_last_game

Play



Each one have a specific role .

Randomword :

```
char * randomword(FILE *fp,int n) {
int      i=0 ;
int      x ;
char    *ch=(char *)malloc((n)*sizeof(char)) ; ;
char    *s=(char *)malloc((n)*sizeof(char)) ; ;
        srand(time(NULL)) ;
        while (fgets(s,100,fp) ) {      /* fgets(s,
        if (strlen(s)==n+1) {                  /* se
                                              if the
                                               but if
                                               that w
        x=rand()%2 ;
        if (x==1) {
            return s ;
        }
        else {
            strcpy(ch,s) ;

        }
        }
    }
    return ch ;
}
```


fgets(s,100,fp) will give us every word in every line because we are sure that there is no word longer than 100.

Now we search the first word with length n then we choose a random number between 1 and 0 .

if the number is 1 we choose that word .

else we search for an other word with the same length . and we repeat the same process.

but if the number is 0 and there is no other word with length n in the list so we have to choose that word that`s why we have to save the content of s in ch every time .

input parameters : the file fp and the length of the word n

the function will return a random word .

Notify

```
int notify(char *s,char c) {
    int i;
    for(i=0;i<strlen(s);i++) {
        if (s[i]==c) {
            return 1 ;
        }
    }
    return 0 ;
}
```


This function is responsible for searching the letter c in the word s

If c exist in s then this function will return 1 else it will return 0

The main role of this function is to tell the user about a previous right guess. That means if the user enters a correct letter which he had entered before. this function will notify him.

Input parameters: the random word s

The guessed letter c

Output : 1 or 0 depending on the existence of c in s .

Modif:

```
char * modif(char* s,char*ch ,char c) {
    char *a=(char *)malloc((strlen(ch))*sizeof(char)) ;
    char *ch1=(char *)malloc((strlen(ch))*sizeof(char)) ;
    strcpy(ch1,s) ;
    int i=0;
    a=strchr(ch,c) ;
    while (  i<strlen(ch)   && a!="\0" ) { /* if the caract
                                              to that caract
                                              so there is nc

        a=strchr(ch,c) ;
        if (ch[i]==c) {
            ch1[i]=c ;  }
            i++ ;
         }
         return ch1 ;
}
```

This function is responsible about the modification in s

I used ch1 because I have to keep the content of s to compare it with s1 in the play function.

If a==\0 then no need to do any modification because a(the guessed letter) doesn`t exist in ch(the random word)

Now if the character c occurs in the string ch more than once we have to change every dash in the string s to that character . After changing the first dash to that character we have to see if a=! "\0" ... if not so there is no other character c in that string so it`s the end of while.

Input parameters : the word with dashes s

The random word ch

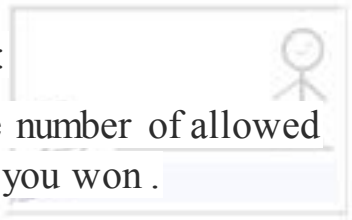
The guessed letter c

Output : this function will return the modified word s containing dashes and the guessed letters.

Show result and time :

```
void show_result_and_time(int d , int x , char* ch,clock_t t) {
    d==x ?  printf("you lost the word is %s\n",ch) : printf("you won\n") ;
    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; // in seconds
    printf("the player took %f seconds to finish the game \n", time_taken);

    save_last_game(time_taken,d-x,ch) ;
}
```



This function will show us the result of each game :

If x(the number of wrong guesses) is equal to d (the number of allowed wrong guesses) that means you lost the game . else you won .

This function will show us also how much time you took to finish the game .

In this function we have also the function save_last_game which is responsible for saving every single game in a file .

Inputs : the number of allowed wrong guesses d

The number of wrong guesses x

The clock t

Outputs : the result of the game and how much time you took to finish it .

Save last game:

```
void save_last_game(double t, int r,char *ch) {
        FILE *fp= fopen("history.txt","a") ;
    printf("give your name please: ") ;
     char s[20] ;
     scanf("%s",s) ;
    if (r==0) {
        fprintf(fp,"%s you took %f to finish the game !! the word was %s ! unfortunately you lost",s,t,ch) ;

        }
        else {
            fprintf(fp,"%s you took %f to guess the word %s ! congratulations you won",s,t,ch) ; }
        fprintf(fp,"\n") ;
}
```

This function is responsible of saving the results of each game in a file called history.

R is the difference between x (the number of wrong guesses) and d(the number of allowed wrong guesses) if it is 0 that means you lost the game .

The user is asked to give his name after the game .

Then the results are saved with his name in the file history.

Inputs : the time taken by the user to finish the game : t

The difference between x and d : R

The random word : ch

Output : there is no input . this function will save the results in the file history.

Play :

```
void play(char*ch,char*s,int d)     {
     char c ;
     int x=0 ;
     char *s1=(char *)malloc((strlen(ch))*sizeof(char)) ;
     clock_t t;
     t = clock();


         while ( x<d && strcmp(s,ch)!=0)  {
        fflush(stdin);
        printf("give a letter :");
        scanf("%c",&c) ;


        if (notify(s,c)==1 ) {
            printf("be careful this guessed letter has already been guessed.\n") ;

         }
         else {
         s1=modif(s,ch,c) ; /* every time i will compare s1 to s if there are differences so tha
                                  if it is true so x is increased by 1 .
                                   if x reached d after several guesses that`s mean that the p

                                    */
       if (strcmp(s1,s) ==0) {
            printf("%s misses %c  //The number of possible guesses are %d\n",s,c,d-x-1) ;
            x++ ;
         }
         else {
            printf("%s correct  guess\n",s1) ;
         }
         strcpy(s,s1) ; }
}

 show_result_and_time(d,x,ch,t) ;
```

This function is responsible of the process of playing .

We can see that it contains many subfunctions. each one have a specific role .

Every time the user is asked to give a letter . then the function notify is responsible about checking whether the given letter has been already guessed and exist in the word or no .

If notify==0 then modif will return the new word with the letter instead of the dash depending on whether the given letter exist in the random word or no .

Then we will compare s1 and s . s1 is the new word after modification

if the given letter exist in the random word then s1 will contain that letter instead of the dash . else s1 will only contain the dashes and it will be equal to s . so we can understand that the given letter doesn`t exist in the word . I will give examples to make everything clear.

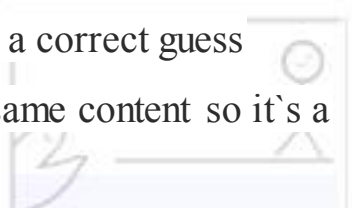
for example the random word is able and the given letter is b :

we have s=----

b exists in the random word so s1 will receive -b—

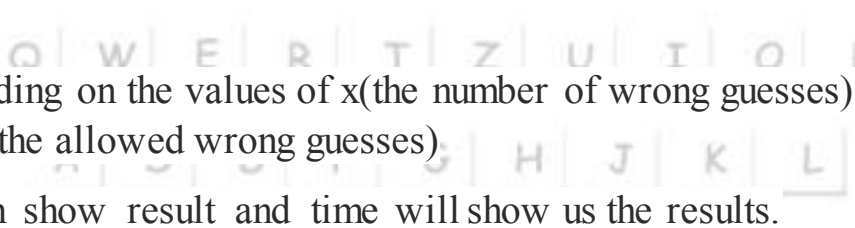
now we compare s and s1 they are different, so it is a correct guess

but if the given letter is z the s and s1 contains the same content so it`s a wrong guess .



after that we have to put the content of s1 in s . and repeat the same process

so now we only have to guess three letters because s contain -b-- ……



then depending on the values of x(the number of wrong guesses) and d(the number of the allowed wrong guesses)

the function show_result_and_time will show us the results.

the input parameters are: the random word Ch.

the word with dashes s

the number of allowed wrong guesses d

output

the result of each try if the guessed letter it true then you will see

correct guess else you will see wrong guess.

The main function :

```
int main ()   {
    int i,x,d ;
    int l ;
    char c;
    char *ch=(char *)malloc((l)*sizeof(char)) ;
    char *s=(char *)malloc((l)*sizeof(char)) ;


    printf("give the length of the word\n") ;
    scanf("%d",&l) ;


    FILE *fp= fopen("words.txt","r") ;

    ch=randomword(fp,l) ;
    ch[l]='\0';

    for (i=0;i<l;i++) {
        s[i]='-' ;
    }
    s[i]='\0';
    puts(s) ;

    printf("change difficulty\n") ;
    scanf("%d",&d) ;

    play(ch,s,d,l) ;




    return 0 ;
}
```

In the main function we can see that everything is clear.

The user is asked first to give a number which is the length of the word.

Then the function randomword will generate a random letter for the file words.

Then we create another word contains dashes with the same length of the random word.

The user is asked to choose the level of difficulty that means the number of allowed wrong letters.

Then the function play will allow us to play.