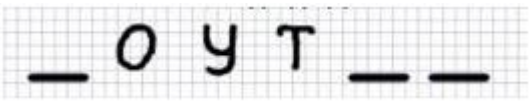
AHORCADOS



Documentos de texto:

Categorías: Archivos de texto plano que almacenan las palabras según su categoría: “Adjetivo.txt”, “Alimento.txt”, “Apellido.txt”, “Lugar.txt”, “Nombre.txt”,“ Objeto.txt”, “Profesión.txt”, “Ser vivo.txt”, “Verbo.txt”.

Registro del puntaje máximo: “Puntaje_maximo.txt”

Dibujo del “ahorcado”: “Dibujo.txt”

Funciones:

Leer de un flujo y escribir en una cadena:

```
char* Leer_F_Escribir_Ca(istream&Categoria,char* x){
            Categoria>>x;
            return x;
};
```

Esta función toma la información contenida en un flujo de entrada (como por ejemplo el teclado (cin) o un archivo de texto) y almacena sus caracteres en una cadena, uno en cada posición de la cadena, hasta encontrar un carácter tal como un espacio, una tabulación, un salto de línea o un fin de texto, por ejemplo.

Crear una matriz:

```
char** Crear_Matriz(int n,int m){
    char** Matriz=new char*[n];
    for (int i=0;i<n;i++){
        Matriz[i]=new char[m];
    };
    return Matriz;
};
```

Esta función toma dos enteros, “n” y “m” en este caso, y los usa para crear una matriz de “n”filas y “m” columnas, creando un vector vertical de tamaño “n” y llenando cada una de sus posiciones de un vector horizontal de tamaño “m” a través de un ciclo.

Seleccionar una palabra al azar dentro una categoría definida:

```
char* Palabra_aleotoria(istream&Categoria,char* x,int Letras){
    int z=1+rand()%(50);
    char* w=new char[Letras];
    while(z>0){
            w=Leer_F_Escribir_Ca(Categoria,x);
            z--;
    };
    return w;
};
```

Esta función toma un documento de texto que contiene una determinada categoría de palabras, y a través de un ciclo ejecuta múltiples veces la función “Leer_F_Escribir_Ca” pasando palabra por palabra. El ciclo se ejecuta un número “z” de veces (número entero aleatorio en el intervalo [1, 50]; 50 es la cantidad de palabras que contiene cada categoría) para finalmente retornar una cadena que contiene una palabra.

Dibujo del ahorcado:

```
//Llenar a la matriz vacia con caracteres del dibujo.
char** Dibujar (char** Matriz_vacia,char** Matriz_dibujo,int s,int v){
    if(s==1){switch (v){
                case 1:Matriz_vacia[1][4]=Matriz_dibujo[1][4];
                break;
                case 2:Matriz_vacia[2][4]=Matriz_dibujo[2][4];
                break;
                case 3:Matriz_vacia[3][4]=Matriz_dibujo[3][4];
                break;
                case 4:Matriz_vacia[2][3]=Matriz_dibujo[2][3];
                break;

                                    (...)

if(s==2){switch (v){
            case 2:Matriz_vacia[1][4]=Matriz_dibujo[1][4];Matriz_vacia[2][4]=Matriz_dibujo[2][4];
            break;
            case 4:Matriz_vacia[3][4]=Matriz_dibujo[3][4];Matriz_vacia[2][3]=Matriz_dibujo[2][3];
            break;
            case 6:Matriz_vacia[2][5]=Matriz_dibujo[2][5];Matriz_vacia[3][3]=Matriz_dibujo[3][3];
            break;
            case 8:Matriz_vacia[3][5]=Matriz_dibujo[3][5];Matriz_vacia[4][6]=Matriz_dibujo[4][6];
            break;
            case 10:Matriz_vacia[4][5]=Matriz_dibujo[4][5];Matriz_vacia[4][4]=Matriz_dibujo[4][4];
            break;
            case 12:Matriz_vacia[4][3]=Matriz_dibujo[4][3];Matriz_vacia[4][2]=Matriz_dibujo[4][2];

                                    (...)

if(s==3){switch (v){
            case 3:Matriz_vacia[1][4]=Matriz_dibujo[1][4];Matriz_vacia[2][4]=Matriz_dibujo[2][4];Matriz_vacia[3][4]=Matriz_dibujo[3][4];
            break;
            case 6:Matriz_vacia[2][3]=Matriz_dibujo[2][3];Matriz_vacia[2][5]=Matriz_dibujo[2][5];Matriz_vacia[3][3]=Matriz_dibujo[3][3];
            break;
            case 9:Matriz_vacia[3][5]=Matriz_dibujo[3][5];Matriz_vacia[4][6]=Matriz_dibujo[4][6];Matriz_vacia[4][5]=Matriz_dibujo[4][5];
            break;
            case 12:Matriz_vacia[4][4]=Matriz_dibujo[4][4];Matriz_vacia[4][3]=Matriz_dibujo[4][3];Matriz_vacia[4][2]=Matriz_dibujo[4][2];
            break;
            case 15:Matriz_vacia[4][1]=Matriz_dibujo[4][1];Matriz_vacia[4][0]=Matriz_dibujo[4][0];Matriz_vacia[3][0]=Matriz_dibujo[3][0];
            break;

                                    (...)
```

Esta es la función responsable del dibujo del “ahorcado” que aparece en pantalla. Dependiendo del nivel de dificultad y de la cantidad de errores cometidos, esta función toma una o varias posiciones específicas de una matriz base (“Matriz_vacía”) y sobrescribe sus

espacios vacíos con caracteres en las mismas posiciones relativas de otra matriz (“Matriz_dibujo”(matriz que contiene al dibujo del “ahorcado”)) .

Principales variables en el “int main( )”:

“Seguir_jugando”: Le permite al usuario definir si se repite o termina el juego.

“y”: Número correspondiente a la Categoría.

“Letras”: Tamaño máximo definido para la cadena que almacena la palabra seleccionada.

“n”: Cantidad de letras en la palabra seleccionada.

“c”: Variable booleana usada repetidas veces para controlar ciclos o obtener a algún tipo de información de ellos.

“j, i”: Variables enteras usadas repetidas veces como índices de cadenas o matrices.

“s_c, s”: Nivel de dificultad.

“b”: Cantidad de aciertos.

“v”: Cantidad de equivocaciones.

“Vocales”: Cantidad de vocales.

“Letra”: Carácter introducido por el usuario.

Procesos en el “int main( )” :

Selección de una categoría y una palabra al azar:

```
int y=1+rand()%(9);
int Letras=30;
char* Palabra=new char[Letras];

    //Selección de una categoría y una palabra al azar.
switch (y){
    case 1:Palabra_aleotoria(Adjetivo, Palabra, Letras);
        break;
    case 2:Palabra_aleotoria(Alimento, Palabra, Letras);
        break;
    case 3:Palabra_aleotoria(Apellido, Palabra, Letras);
        break;
    case 4:Palabra_aleotoria(Lugar, Palabra, Letras);
        break;
    case 5:Palabra_aleotoria(Nombre, Palabra, Letras);
        break;
    case 6:Palabra_aleotoria(Objeto, Palabra, Letras);
        break;
    case 7:Palabra_aleotoria(Profesion, Palabra, Letras);
        break;
    case 8:Palabra_aleotoria(Ser_vivo, Palabra, Letras);
        break;
    case 9:Palabra_aleotoria(Verbo, Palabra, Letras);
        break;

};
```

Este conmutador se vale de la variable aleatoria “y” para seleccionar una de las nueve categorías de palabras, y a su vez aplica la función “Palabra_aleatoria” a la cadena vacía “Palabra”, para llenarla con una palabra seleccionada al azar dentro de dicha categoría.

Determinación de la cantidad de letras:

```
int n=0;
bool c=0;

    //Determinacion de la cantidad de letras.
while(c==0){
  if ((int)(Palabra[n])==0){c=1;}
  else {n++;};
};
```

Este ciclo verifica cada posición de la cadena “Palabra” (cadena que contiene a la palabra elegida) hasta encontrar el carácter nulo (decimal “0” en el código ASCII). Conforme realiza este proceso, va incrementando a la variable “n” en una unidad. De este modo la variable “n” termina siendo equivalente a la cantidad letras que contiene la palabra.

Construcción de la palabra oculta:

```
char* Palabra_oculta=new char[n+1];
int j=0;

while(j<n){
    Palabra_oculta[j]='_';
    j++;
};
```

Este ciclo toma una nueva cadena llamada “Palabra_oculta” y la llena de tantos rayalpisos (“_”) como cantidad de letras tiene la palabra seleccionada.

Llenado de una matriz con los caracteres del dibujo:

```
char** Matriz_dibujo=Crear_Matriz(5,7);
for (int i=0;i<5;i++){
    for (int j=0;j<7;j++){
        char t;
        Dibujo>>t;
        Matriz_dibujo[i][j]=t;
    };
};
```

Este par de ciclos llenan a la matriz “Matriz_dibujo” casilla por casilla con los caracteres de un archivo de texto plano llamado “Dibujo.txt” (este documento contiene los caracteres del dibujo).

Llenado de una matriz con espacios vacíos (“ ”):

```
char** Matriz_vacia=Crear_Matriz(5,7);
for (int i=0;i<5;i++){
    for (int j=0;j<7;j++){
        Matriz_vacia[i][j]=32;
    };
};
```

Este par de ciclos toman una nueva matriz del mismo tamaño que la anterior, y llenan cada una de sus posiciones con espacios vacíos (decimal “32” en el código ASCII).

Descripción del juego:

```
cout <<"Ahorcados Clasico"<<'\n'<< endl;
cout <<"El juego consiste en adivinar una palabra seleccionada al azar antes"<<endl;
cout <<"de que el dibujo del 'ahorcado' se termine, puede valerse de pistas"<<endl;
cout <<"si lo desea, pero usarlas tambien genera una progresion en el dibujo"<<endl;
cout <<"del 'ahorcado'.Pulse una letra para intentar adivinar la palabra.Pulse"<<endl;
cout <<"la tecla '0' para usar una pista.Advertencia: Las pistas pueden repetirse."<<'\n'<<endl;
```

Esto es simplemente una impresión sucesiva en pantalla (renglón por renglón) de una breve descripción del juego.

Determinación del nivel de dificultad:

```
char s_c;
int s;
cin >>s_c;
if (!(s_c==49||s_c==50||s_c==51)){
    cout <<'\n'<<"El caracter introducido no corresponde a ningun nivel."<<endl;
    cout <<"Se seleccionara por defecto: 'Dificil'."<<endl;
    s=3;
    cout <<"Presione una tecla para continuar."<<endl;
    cin.get();cin.get();
}else {
    if (s_c==49){s=1;};
    if (s_c==50){s=2;};
    if (s_c==51){s=3;};
};
```

Este condicional permite al usuario elegir un nivel antes de empezar el juego. El carácter introducido por el usuario es almacenado en una variable de tipo char (llamada “s_c”) para poder identificar si es uno de los tres caracteres aceptados (“1”, “2” o “3”). Si es uno de dichos tres caracteres (decimales “49”, “50” y “51” respectivamente en el código ASCII) se le asigna a la variable entera “s” el nivel (niveles representados por los números “1”, “2” y “3” (fácil, medio y difícil respectivamente)) correspondiente al carácter introducido por el usuario; de esta forma el programa puede ejecutar las operaciones aritméticas que requieren de este valor, ya que de ser una variable char no podría hacerlo. Si no es ninguno de los tres caracteres válidos el condicional asigna el nivel 3 (difícil) automáticamente.

Desarrollo del juego:

Impresión del dibujo en pantalla:

```
cout <<"Dibujo: "<<'\n'<<endl;
for (int i=0;i<5;i++){
    for (int j=0;j<7;j++){
        cout <<Matriz_vacia[i][j];
    };
cout<<'\n';
};
cout<<'\n';
```

Este par de ciclos imprimen en pantalla cada uno de los caracteres contenidos en la matriz “Matriz_vacía” (matriz que conforme avanza el juego se va llenando de los caracteres del dibujo).

Impresión de las letras usadas en pantalla:

```
j=0;
cout <<"Letras usadas: ";
while (Letras_usadas[j]!=0){
    cout <<Letras_usadas[j]<<" ";
    j++;
};
cout<<'\n'<<endl;
```

Este ciclo imprime en pantalla cada uno de los caracteres almacenados en la cadena "Letras_usadas" (cadena que almacena las letras que el usuario ya ha ingresado dentro de sí), y detrás de cada uno imprime un espacio para separarlos.

Impresión de la palabra oculta en pantalla:

```
cout <<"Palabra: "<<'\n'<<endl;

    //Impresion en la pantalla de la palabra oculta.
j=0;
while (j<n){
    cout <<Palabra_oculta[j]<<" ";
    j++;
};
cout<<'\n'<<endl;
```

Este ciclo imprime en pantalla cada uno de los caracteres de la cadena "Palabra_oculta", seguido de un espacio espacio para separarlos.

Solicitud de una Letra al usuario:

```
char Letra;
cin>>Letra;
cout<<'\n'<<endl;
```

Esto simplemente le permite al usuario ingresar caracteres al programa.

Conversión de mayúsculas a minúsculas:

```
if ((int)Letra>=65&&(int)Letra<=90){
    Letra=(int)Letra+32;
};
```

Si el usuario ingresa una letra mayúscula (las mayúsculas están entre los decimales "65" y "90" en el código ASCII), el programa modifica la variable char que contiene dicho carácter, reemplazandolo con aquel que está a 32 unidades en el código ASCII. De esta forma se cambia a una letra mayúscula por su correspondiente en minúscula.

Verificación de que el carácter introducido es una letra:

```
if (!((int)Letra>=97&&(int)Letra<=122)&&(Letra!='0')){
    cout <<"El caracter introducido no es una letra"<<endl;
    cout <<'\n'<<"Presiona una tecla para continuar.";
    cin.get();cin.get();
    v=v+s;
    Matriz_vacia=Dibujar(Matriz_vacia,Matriz_dibujo,s,v);
```

Este condicional verifica que el carácter almacenado en la variable "Letra" es una letra minúscula (las minúsculas están entre los decimales 97 y 122 en el código ASCII) o el carácter "0". En caso de no serlo, se lo informa al usuario en pantalla, e incrementa el número de errores (en una, dos o tres unidades dependiendo del nivel), para luego aumentar la evolución del dibujo del ahorcado (al aplicar la función "Dibujar" a la matriz "Matriz_vacía").

Pistas:

```
}else if (Letra=='0'){
    cout <<'\n'<<"Pista: ";
    int z=1+rand()%(3);
    Vocales=0;
    switch (z){
```

Este condicional permite al usuario solicitar pistas si introduce el caracter "0".Al hacerlo se crea una variable aleatoria ("z": variable entera aleatoria en el intervalo [1,3]) , y con ella, por medio de un conmutador, se selecciona una de tres pistas disponibles.

- Cantidad de vocales:

```
case 1:j=0;
    while (j<n){
        if (Palabra[j]=='a'||Palabra[j]=='e'||Palabra[j]=='i'||Palabra[j]=='o'||Palabra[j]=='u'){Vocales++;};
        j++;
    };
    cout <<"La palabra tiene "<<Vocales<<" vocales."<<endl;
break;
```

Este ciclo verifica, casilla por casilla, la cadena "Palabra", en busca de una letra que sea vocal. En caso de serlo, incrementa a la variable "Vocales" en una unidad. De este modo la variable "Vocales" termina siendo equivalente a la cantidad de vocales que posee la palabra.

- Cantidad de consonantes:

```
case 2:j=0;
    while (j<n){
        if (Palabra[j]=='a'||Palabra[j]=='e'||Palabra[j]=='i'||Palabra[j]=='o'||Palabra[j]=='u'){Vocales++;};
        j++;
    };
    cout <<"La palabra tiene "<<n-Vocales<<" consonantes."<<endl;
break;
```

Este ciclo hace exactamente lo mismo que el anterior. La diferencia está en el momento de imprimir en pantalla, pues en vez de imprimir lo que está contenido en "vocales" (cantidad de vocales) imprime la diferencia entre "n" y "vocales" (cantidad de letras que posee la palabra menos cantidad de vocales) que corresponde a la cantidad de consonantes que posee la palabra.

- Categoría:

```
    case 3:switch (y){
            case 1:cout<<"Es un adjetivo."<<endl;
            break;
            case 2:cout<<"Es un alimento."<<endl;
            break;
            case 3:cout<<"Es un apellido."<<endl;
            break;
            case 4:cout<<"Es un lugar."<<endl;
            break;
            case 5:cout<<"Es un nombre."<<endl;
            break;
            case 6:cout<<"Es un objeto."<<endl;
            break;
            case 7:cout<<"Es una profesion."<<endl;
            break;
            case 8:cout<<"Es un ser vivo."<<endl;
            break;
            case 9:cout<<"Es un verbo."<<endl;
            break;

        };
    break;
};
```

Este conmutador retoma a la variable "y" (variable aleatoria usada anteriormente para determinar la categoría) para elegir una de las nueve descripciones como pista para el usuario (una por cada categoría de palabra).

Comprobación de que la letra introducida no había sido introducida antes:

```
c=0;
j=0;
bool d=0;

    //comprobación de que la letra introducida no había sido introducida antes.
while (c==0){
    if (Letra==Letras_usadas[j]){c=1;};
    if (Letras_usadas[j]==0){c=1;d=1;};
    j++;
};

if (d==0){
cout <<"Ya introdujo esa letra, introduzca una diferente"<<endl;
cout <<'\n'<<"Presiona una tecla para continuar.";
cin.get();cin.get();
```

Este ciclo revisa posición por posición a la cadena "Letras_usadas" y compara cada una con la letra introducida por el usuario. El ciclo termina si en algún momento la letra introducida

por el usuario y la letra que se encuentra en una determinada posición son las mismas; o si llega al final de la cadena (al encontrar el carácter nulo (decimal "0" en el código ASCII)). La variable booleana "d" es modificada sólo si el ciclo llega al final de la cadena, de este modo y con el condicional siguiente, se le informa al usuario que la letra que introdujo ya había sido introducida antes, en caso de que el ciclo nunca llegue al final de la cadena.

Llenado de la palabra oculta y registro de las letras usadas:

```
j=0;
c=0;
while(j<n){
    if(Letra==Palabra[j]){Palabra_oculta[j]=Letra;b++;c=1;};
    j++;
};
if (c==0){
    v=v+s;
    Matriz_vacia=Dibujar(Matriz_vacia,Matriz_dibujo,s,v);
};
Letras_usadas[k]=Letra;
Letras_usadas[k+1]=0;
k++;
```

Este ciclo compara la letra introducida por el usuario con cada uno de los caracteres pertenecientes a la palabra. En caso de la letra introducida por el usuario y una letra en una determinada posición de la palabra sean la misma, el ciclo almacena dicha letra en la cadena "Palabra_oculta" en la misma posición en la que se encuentra en la palabra; además, solo si esto sucede por lo menos una vez, la variable "c" es modificada; si la variable "c" no es modificada se activa un condicional que incrementa la cantidad de errores (variable "v") y aplica la función "Dibujar" a la matriz "Matriz_vacia" (función que conduce a la evolución del dibujo mostrado en pantalla).

Derrota:

```
if (v==24){
    cout <<"Dibujo: "<<'\n'<<endl;
    for (int i=0;i<5;i++){
        for (int j=0;j<7;j++){
            cout <<Matriz_vacia[i][j];
        };
    cout<<'\n';
    };
    cout<<'\n';
    cout <<"Ahorcado!"<<'\n'<<endl;
    cout <<"Has perdido!"<<'\n'<<endl;
    cout <<"La palabra era: "<<Palabra<<'\n'<<endl;
}
```

Si la cantidad de errores (variable "v") llega a 24 (cantidad de segmentos que componen el dibujo), el juego termina y este condicional se activa, mostrando en pantalla el dibujo completado y la palabra, y anunciando al usuario que ha perdido.

Victoria:

```
if (b==n){
    cout <<"Dibujo: "<<'\n'<<endl;
    for (int i=0;i<5;i++){
        for (int j=0;j<7;j++){
            cout <<Matriz_vacia[i][j];
        };
    cout<<'\n';
    };
    cout<<'\n';
    cout <<"Palabra: "<<Palabra<<'\n'<<endl;
    cout <<"Has ganado!"<<'\n'<<endl;
    double Puntaje=100*s*n/(v+1);
    cout <<"Puntaje: "<<Puntaje<<'\n'<<endl;
    double w;
    Puntaje_maximo>>w;
    if (Puntaje>w){
        cout <<"Nuevo puntaje maximo!"<<'\n'<<endl;
        Puntaje_maximo.close();
        ofstream Puntaje_maximo ("Puntaje_maximo.txt");
        Puntaje_maximo<<Puntaje;
    }
    };
```

Si la cantidad de aciertos (variable “b”) es igual a la cantidad de letras que posee la palabra (variable “n”), el juego termina y este condicional se activa, mostrando en pantalla el dibujo y la palabra, anunciando al usuario que ha ganado y mostrándole su puntaje ((100*nivel*cantidad de letras que contiene la palabra) / (cantidad de errores+1)).