PROGRAMACIÓN APLICADA Y LAB.

Práctica No. 2

Prof. Maestro Jorge Rodríguez

Mauricio de Garay (209667-5)

Bernardo García Ramos (209679-5)

Patricia Martínez (210931-7)

|  |
| --- |
| Universidad Iberoamericana |
|  |

***Resumen*- En este documento está documentada toda la elaboración de la segunda práctica de Programación Aplicada y Laboratorio, en el cual se va a poder observar la definición del problema y diseño de solución (tal como los diagramas IPO y el pseudocódigo).**

1. Descripción del problema

Se requiere un programa que genere los códigos de longitud variable con respecto a una serie de símbolos con su probabilidad que serán dadas por el usuario. Para un mejor funcionamiento del programa, se tendrá un menú el cual constará de las siguientes opciones:

1. Ingresar un símbolo y su probabilidad
2. Listar los símbolos introducidos
3. Eliminar un símbolo
4. Modificar un símbolo
5. Generar los códigos
6. Guardar diccionario
7. Cargar diccionario
8. Codificar mensaje
9. Decodificar mensaje
10. Salir

Para la opción de “Generar los códigos”, los códigos generados se mostrarán al usuario. Así, una vez generado el árbol, este servirá para codificar o decodificar algún mensaje.

1. Comportamiento Detallado

* Nombre del desarrollador
* Mensaje introductorio que explica brevemente el objetivo del sistema
* Menú con sus respectivas opciones:
  + - 1. ***Ingresar símbolo y su probabilidad***
* Se pide símbolo y probabilidad al usuario
* Ir verificando si ya se ha llegado al 100%.
  + - 1. ***Listar los símbolos introducidos***
* Desplegar los símbolos introducidos con su respectiva probabilidad
  + - 1. ***Eliminar un símbolo***
* Pedir el símbolo que se desea eliminar
* Borrar dicho símbolo
  + - 1. ***Modificar un símbolo***
* Pedir el símbolo que se desea modificar
* Modificar la probabilidad del símbolo
  + - 1. ***Generar los códigos***

Si no se tiene un código:

* Verificar que se tenga el 100% de la probabilidad total de los símbolos.
* Generar el árbol con el algoritmo de Greedy

Si ya se tiene un código:

* Codificar mensaje
* Decodificar mensaje
* Guardar mensaje codificado/decodificado en archivo
  + - 1. ***Guardar diccionario***
* Verificar que ya se haya generado algún código
* Pedir el nombre para asignarlo al diccionario
* Guardar diccionario
  + - 1. ***Cargar diccionario***
* Pedir el nombre para buscar el diccionario
* Leer diccionario y cargarlo
  + - 1. ***Salir***
* Se termina el programa

1. Entradas y Salidas

Entrada:

* Símbolo
* Probabilidad
* Diccionario ya generado
* Mensaje codificado/decodificado

Salida:

* Diccionario a generar
* Mensaje codificado/decodificado

1. Procesos

* Despliega mensaje de desarrollador y pide que se ingrese la tecla <Enter>.
* Despliega mensaje explicando el programa y pide que se ingrese la tecla <Enter>.
* Despliega Menú y pide que se elija una opción.
  + Si se ingresa opción 1, se ingresa a “Ingresar símbolo y su probabilidad”
    - Pedir símbolo
    - Pedir probabilidad
    - Verificar que se tenga el 100%
    - Avisar al usuario cuánto le falta/sobra de probabilidades
  + Si se elige opción 2, se ingresa a “Listar los símbolos introducidos”
    - Desplegar todos los símbolos con sus respectivas probabilidades
  + Si se elige opción 3, se ingresa a “Eliminar un símbolo”
    - Pedir símbolo a modificar
    - Buscar el símbolo con los que han sido introducidos
    - Si se encuentra, mostrarlo y preguntar si de verdad se desea borrar.
    - Si se dice que sí, borrar el símbolo con su probabilidad
  + Si elige la opción 4, se ingresa a “Modificar un símbolo”
    - Pedir símbolo a modificar
    - Buscar el símbolo con los que han sido introducidos
    - Si se encuentra, preguntar si de verdad se desea modificar
    - Si se desea modificar, volver a pedir símbolo y probabilidad
  + Si se elige la opción 5, se ingresa a “Generar los códigos”
    - Verificar que se tenga el 100% de los símbolos
    - Generar árbol de código con el algoritmo de Greedy
    - Desplegar todos los símbolos con su respectivo código.
  + Si se ingresa la opción 6, se ingresa a “Guardar diccionario”
    - Verificar que se tenga algún código generado
    - Guardar diccionario
  + Si se ingresa la opción 7, se ingresa a “Cargar diccionario”
    - Verificar que no se tenga algún código generado
    - Cargar diccionario
  + Si se ingresa la opción 8, se ingresa a “Codificar mensaje”
    - Pedir el mensaje que se desea codificar
    - Pedir nombre de archivo en donde se guarde el mensaje codificado
    - Guardar mensaje codificado
  + Si se ingresa la opción 9, se ingresa a “Decodificar mensaje”
    - Pedir nombre de archivo en donde se tiene el mensaje codificado
    - Leer mensaje codificado
    - Desplegar mensaje decodificado
    - Pedir nombre de archivo donde se guarde el mensaje decodificado
    - Guardar mensaje decodificado
  + Si se ingresa la opción 10, se termina el programa

1. Alcances y Limitaciones

Alcances: El programa únicamente cuenta con la capacidad trabajar con las opciones dadas por el menú, tales como: Ingresar símbolo y probabilidad, Listar símbolos introducidos, borrar símbolos, modificar símbolos, generar código, guardar diccionario, leer diccionario, codificar o decodificar algún mensaje.

Limitaciones: El programa no puede codificar o decodificar mensajes si su correspondiente diccionario, de tal forma que si se encuentra algún error, el programa lo indicará y se regresará al menú.

1. Diseño de pantalla
   1. **Primer Escenario – Escenario Principal (Desplegar Menú)**

$practica2.c

Desarrolladores: Mauricio De Garay, Bernardo García Ramos y Patty Martínez

Presione la tecla <Enter> para continuar:

Este programa está diseñado para generar códigos de longitud variable con respecto a una serie de símbolos con su probabilidad que serán dadas por usted, el usuario. Se podrán ingresar símbolos con sus probabilidades, listar los mismos, modificar o borrar alguno, generar el código, leer diccionario, cargar diccionario, codificar o decodificar algún mensaje. ¡Disfrute el programa!

Presione la tecla <Enter> para continuar:

PRÁCTICA 2

MENÚ:

1. Ingresar un símbolo y su probabilidad
2. Listar los símbolos introducidos
3. Eliminar un símbolo
4. Modificar un símbolo
5. Generar los códigos
6. Guardar diccionario
7. Cargar diccionario
8. Codificar mensaje
9. Decodificar mensaje
10. Salir

Ingrese una opción: 10

HASTA PRONTO

* 1. **Segundo Escenario (Ingresar símbolos)**

Dame el símbolo: A

Dame su probabilidad: 20

Presiona <Enter> para regresar al menú:

* 1. **Tercer Escenario (Listar símbolos)**

Los símbolos introducidos son los siguientes:

A – 20%

D – 42 %

Z – 10%

Falta un 28% para llegar al 100%.

Presiona <Enter> para regresar al menú:

* 1. **Cuarto Escenario (Borrar símbolos)**

Dame el símbolo que deseas eliminar: D

Se ha encontrado el siguiente símbolo: D – 42%. Presiona 1 si deseas eliminarlo: 1

Se ha eliminado exitosamente.

Presiona <Enter> para regresar al menú:

* 1. **Quinto Escenario (Modificar símbolos)**

Dame el símbolo que deseas modificar: D

Se ha encontrado el siguiente símbolo: D – 42%. Presiona 1 si deseas modificarlo: 1

Dame el nuevo símbolo: G

Dame su nueva probabilidad: 5

Presiona <Enter> para regresar al menú:

* 1. **Sexto Escenario (Generar códigos) / Códigos inventados**

Ya se tiene el 100% de los símbolos dados.

Se van a generar os códigos…

Los códigos son los siguientes:

A – 010

D – 1

Z - 011

Presiona <Enter> para regresar al menú:

* 1. **Séptimo Escenario (Guardar diccionario)**

Dame el nombre del archivo donde deseas guardar el diccionario generado: diccio1

Se ha guardado el diccionario exitosamente.

Presiona <Enter> para regresar al menú:

* 1. **Octavo Escenario (Cargar diccionario)**

Dame el nombre del archivo donde está guardado el diccionario generado: diccio1

Se ha cargado el diccionario exitosamente.

Presiona <Enter> para regresar al menú:

* 1. **Noveno Escenario (Codificar mensaje) // Asumiendo que se tienen todos los símbolos**

Escribe a continuación el mensaje que deseas codificar: Hola. Esto es un mensaje prueba.

Dame el nombre del archivo en donde deseas guardar el mensaje codificado: mensaje1cod

Se ha guardado y codificado exitosamente.

Presiona <Enter> para regresar al menú:

* 1. **Décimo Escenario (Decodificar mensaje)**

Dame el nombre del archivo en donde está guardado el mensaje codificado: mensaje1

El mensaje decodificado es: Hola. Esto es un mensaje prueba.

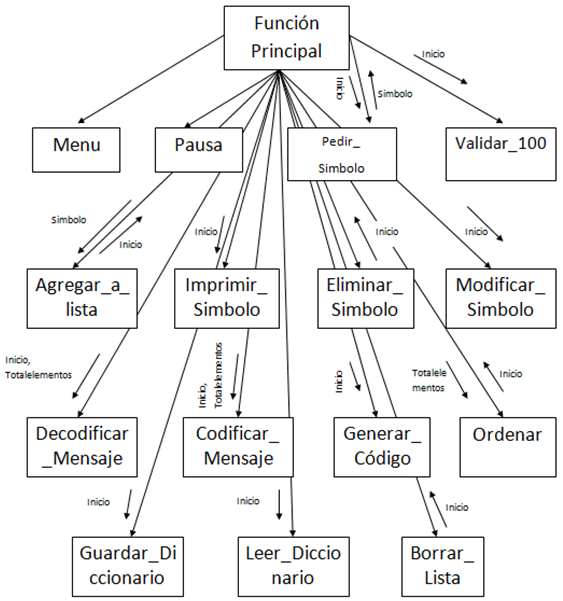
Dame el nombre del archivo en donde deseas guardar el mensaje decodificado: mensaje1dec

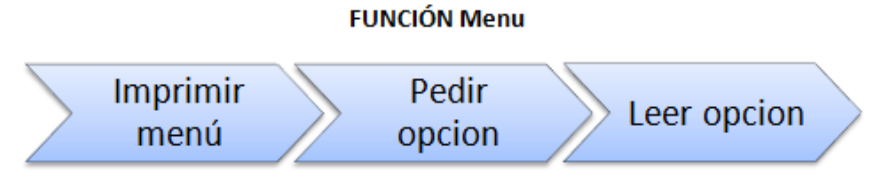
Se ha guardado y codificado exitosamente.

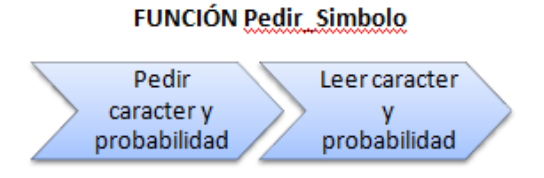
Presiona <Enter> para regresar al menú:

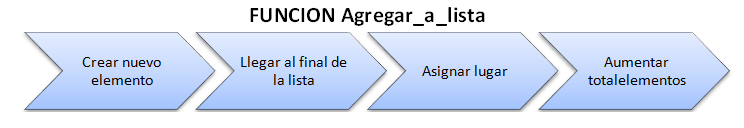
1. Diseño de Solución

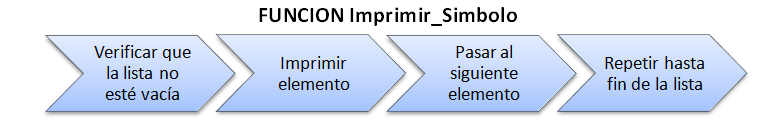
Diagramas IPO y pseudocódigo anexados al final.

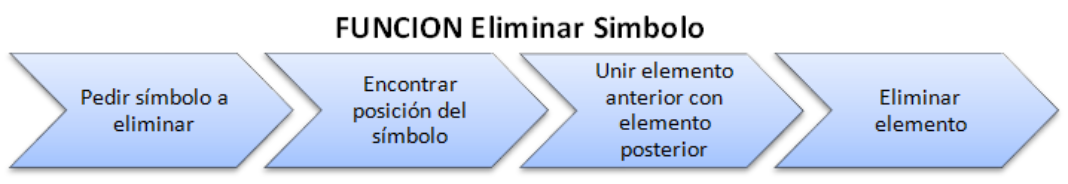
DIAGRAMAS IPO 

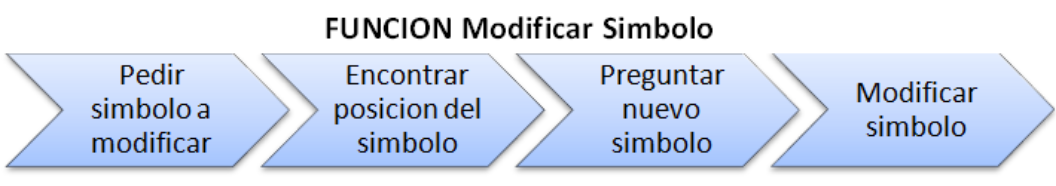


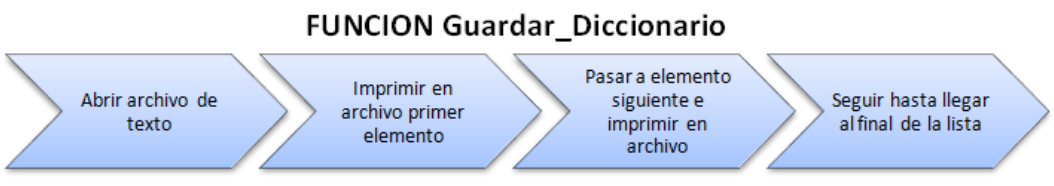


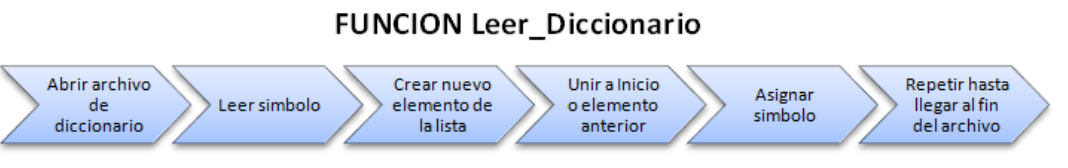


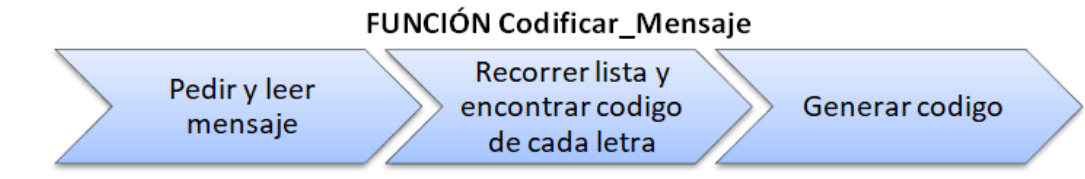
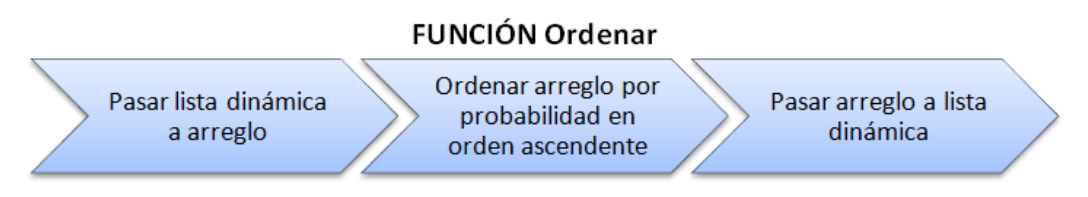


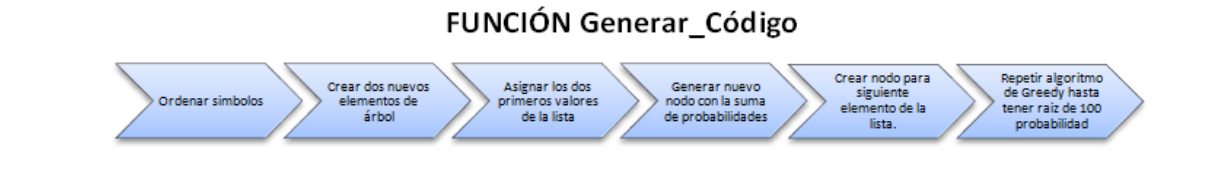


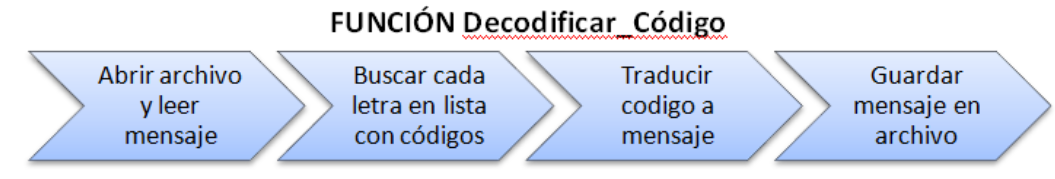


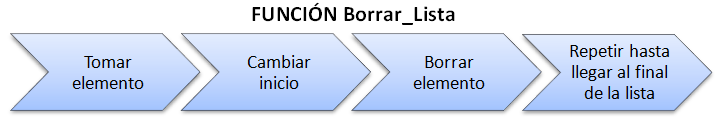












PSEUDOCÓDIGO:

PROGRAMA

FUNCION PRINCIPAL ( )

INICIO

Correcto=0;

Totalelementos, Inicio=NULL;

System(“clear”);

Write(|”Este programa fue creado por Mauricio de Garay, Bernardo García Ramoz y Patricia Martinez.”);

Write(|”Bienvenido a codificación y decodificación de mensajes.”);

Write(|”Para poder codificar o decodificar un mensaje, se tendrá que ingresar un diccionario que incluya cada símbolo y

su probabilidad de aparición.”);

Pausa();

System(“clear”);

DO

INICIO

Opcion=Menu();

SWITCH(Opcion)

INICIO

Caso ‘1’:

Pedir\_Simbolo(Inicio | &Simbolo);

Agregar\_a\_lista( Simbolo | &Inicio);

Totalelementos=Totalelementos+1;

Validacion=Validar\_100(Inicio);

IF(Validacion>100) THEN

INICIO

Write( | “Tus probabilidades exceden el 100% por un %d%. Revisar y modificar.”| Validacion);

FIN

IF(Validacion<100) THEN

INICIO

Write( | “Tus probabilidades les falta el %d% para llegar al 100%.”| Validacion-100);

FIN

Fin;

Caso ‘2’:

Imprimir\_Simbolos(Inicio);

Fin;

Caso ‘3’:

Eliminar\_Simbolo(&Inicio, &Totalelementos);

Write(|”Tu nueva lista: “);

Imprimir\_Simbolos(Inicio);

Fin;

Caso ‘4’:

Modificar\_Simbolo(Inicio);

Validacion=Validar\_100(Inicio);

IF(Validacion>100) THEN

INICIO

Write( | “Tus probabilidades exceden el 100% por un %d%. Revisar y modificar.”| Validacion-100);

FIN

IF(Validacion<100) THEN

INICIO

Write( | “Tus probabilidades les falta el %d% para llegar al 100%.”| Validacion-100);

FIN

Fin;

Caso ‘5’:

Validacion=Validar\_100(Inicio);

IF(Validacion!=100)THEN

INICIO

Write( | “Tus probabilidades no dan 100%. Error.”);

FIN

Raiz

ELSE

INICIO

Ordenar(Totalelementos, &Inicio);

Generar\_Arbol(Inicio, &Raiz);

Temp=Raiz;

Generar\_Codigo( Inicio, Raiz, Temp |);

Borrar\_Arbol(&Raiz);

Correcto=1;

FIN

Fin;

Caso ‘6’:

Guardar\_Diccionario(Inicio|Totalelementos);

Fin;

Caso ‘7’:

Borrar\_Lista(|Inicio);

Inicio=NULL;

Leer\_Diccionario(|Inicio,Totalelementos);

Fin;

Caso ‘8’:

IF(Correcto==1)THEN

INICIO

Codificar\_Mensaje(Inicio, Totalelementos | );

FIN

ELSE

INICIO

Write( | “No has generado el código”);

FIN

Fin;

Caso ‘9’:

IF(Correcto==1)THEN

INICIO

Decodificar\_Mensaje(Inicio, Totalelementos | );

FIN

ELSE

INICIO

Write( | “No has generado el código”);

FIN

Fin;

Caso ‘10’:

Borrar\_Lista(|Inicio);

return 0;

Fin;

default:

Write( | “Dame opciones validas.”);

Fin;

FIN

FIN

WHILE(Opcion!=’10’);

FIN

Menu(|)

INICIO

Write( | “Menú: “);

Write( | “1.- Ingresar símbolo y su probabilidad.¨);

Write( | “2.-Listar símbolos introducidos.”);

Write( | “3.- Eliminar un símbolo. “);

Write( | “4.- Modificar un símbolo.¨);

Write( | “5.- Generar códigos.”);

Write( | “6.- Guardar diccionario en un archivo.¨);

Write( | “7.- Cargar diccionario de un archivo.”);

Write( | “8.- Codificar mensaje en archivo. “);

Write( | “9.- Decodificar mensaje de archivo.¨);

Write( | “10.- Salir..”);

Read( &opc | );

Return opc;

FIN

Pausa(|)

INICIO

Write(|“Presiona <enter> para continuar…. “);

Read( &cont | );

FIN

Pedir\_Simbolo(Inicio | Simbolo)

INICIO

Repeticion=0;

DO

INICIO

IF(Repeticion==1)

INICIO

Write(“Ya habias escrito este caracter.”);

FIN

Repeticion=0;

temp=Inicio;

Write(|”Dame tu simbolo: “);

Read( &Simbolo.Caracter | );

WHILE(temp->sig!=NULL)

INICIO

IF(temp->Caracter==Simbolo.Caracter)

INICIO

Repeticion=1;

FIN

temp=temp->sig;

FIN

FIN

WHILE(Repeticion==1);

Write(“Dame su probabilidad de aparición: “);

Read( &Simbolo.Probabilidad | );

FIN

Validar\_100(Inicio | )

INICIO

Total\_Porcentaje=0;

Temp=Inicio;

WHILE(Temp->sig!=NULL)

INICIO

Total\_Porcentaje=Total\_Porcentaje+Temp->Probabilidad;

Temp=Temp->sig;

FIN

IF(Total\_Porcentaje==100) THEN

INICIO

Return 100;

FIN

IF(Total\_Porcentaje<100) THEN

INICIO

Return Total\_Porcentaje;

FIN

IF(Total\_Porcentaje>100) THEN

INICIO

Return Total\_Porcentaje;

FIN

FIN

Agregar\_a\_lista(Simbolo | Inicio)

INICIO

temp=(tSimbolo\*)malloc(sizeof(tSimbolo));

temp->Caracter=Simbolo.Caracter;

temp->Probabilidad=Simbolo.Probabilidad;

IF(\*Inicio==NULL)THEN

INICIO

temp->sig=NULL;

\*Inicio=temp;

FIN

ELSE

INICIO

temp2=\*Inicio;

WHILE(temp2->sig!=NULL)

INICIO

temp2=temp2->sig;

FIN

temp2->sig=temp;

temp->sig=NULL;

FIN

FIN

Imprimir\_Simbolos(Inicio | )

INICIO

IF(INICIO==NULL)THEN

INICIO

Write(| “La lista esta vacia.”);

return;

FIN

temp=INICIO;

WHILE(temp!=NULL)

INICIO

Write(1”Simbolo: Probabilidad: “, temp->Caracter, temp->Probabilidad);

temp=temp->sig;

FIN

FIN

Eliminar\_Simbolo( |Inicio)

INICIO

correcto=0;

Write( | “Dame el simbolo que deseas eliminar: “);

Read( &Elim | );

IF(\*Inicio==NULL)THEN

INICIO

printf("ERROR: La lista esta vacia.\n");

return 0;

FIN

temp=\*Inicio;

IF(temp->sig==NULL && temp->Caracter==Elim)THEN

INICIO

Write("Se ha encontrado el simbolo y sera eliminado. Presiona <enter> para continuar... ");

Read( &continuar|);

\*Inicio=NULL;

\*Totalelementos=0;

free(temp);

return 0;

FIN

temp2=\*Inicio;

WHILE(temp->sig!=NULL)

INICIO

temp2=temp;

temp3=temp->sig;

IF(temp==\*Inicio && temp->Caracter==Elim)THEN

INICIO

Write("Se ha encontrado el simbolo y sera eliminado. Presiona <enter> para continuar... ");

Read( &continuar|);

\*Inicio=(\*Inicio)->sig;

Totalelemento=Totalelemento-1;

free(temp);

return 0;

FIN

IF(temp3->Caracter==Elim)THEN

INICIO

Write(|"Se ha encontrado el simbolo y sera eliminado. Presiona <enter> para continuar... ");

Read(&continuar|);

temp2->sig=temp3->sig;

free(temp3);

Totalelemento=Totalelemento-1;

correcto=1;

return 0;

FIN

temp=temp->sig;

FIN

IF(correcto==0)THEN

INICIO

Write(|"ERROR: No se encontro ese simbolo en la lista\n");

return 0;

FIN

FIN

Modificar\_Simbolo(Inicio | )

INICIO

Write( | “Dame el simbolo que deseas modificar: “);

Read( &Mod | );

temp=Inicio;

correcto=0;

WHILE(temp->sig!=NULL)

INICIO

IF(temp->Caracter==Mod)THEN

INICIO

correcto=1;

Write( |”Se ha encontrado el simbolo. Dime su nueva probabilidad: “);

Read( &Nueva | );

temp->Probabilidad=Nueva;

FIN

temp=temp->sig;

FIN

IF(correcto==0)THEN

INICIO

Write( |”No se encontro el simbolo en tu diccionario.”);

FIN

FIN

FUNCIÓN Decodificar\_Mensaje (Inicio, Diccionario, Long | )

INICIO

Variables: Nombre\_de\_Archivo, Guardar\_Archivo, Leido, Temp, i=0, BORRAR;

WRITE(“Dame el nombre del archivo que contiene el mensaje a decodificar: “);

READ(|&Nombre\_de\_Archivo);

IF(OPEN==ARCHIVO(Nombre\_de\_Archivo))

INICIO

WHILE(!feof)

INICIO

READ(Archivo, &Letra);

Temp=Inicio;

WHILE(i<Long)

INICIO

WHILE(Temp!=NULL)

INICIO

IF(Temp->Código[i]!=Letra) THEN

INICIO

Temp->Bandera=0;

FIN

IF(Temp->Código[i]==Letra) THEN

INICIO

IF(Temp->Código[i+1]==NULL) THEN

INICIO

IF(Temp->Bandera==1) THEN

INICIO

COPY(Temp->Caracter, Mensaje);

Temp=NULL;

i=Long;

FIN

FIN

FIN

Temp=Temp->Next;

FIN

i=i+1;

FIN

Temp = Inicio;

WHILE(Temp!=NULL)

INICIO

IF(Temp->Bandera==1)

INICIO

ERROR=1;

FIN

FIN

IF(ERROR==0)

INICIO

WRITE(“ERROR TOTAL: NO SE CUENTA CON LOS DATOS PARA DECODIFICAR EL MENSAJE.”);

EXIT(0);

FIN

ERROR=0;

i=i+1;

IF(i>Long)

INICIO

i=0;

temp=Inicio;

WHILE(Temp!=NULL)

Temp->Bandera=1;

FIN

FIN

WRITE(“El mensaje decodificado es: %s\n”, Mensaje);

WRITE(“Dame el nombre del archivo donde lo deseas guardar: “);

READ(“%d”, &Guardar\_Archivo);

OPEN(Archivo(Guardar\_Archivo))

WRITE(Archivo, “%s”, Mensaje);

CLOSE(Archivo);

CLOSE(Guardar\_Archivo);

WRITE(“Se guardó exitosamente. Presiona <Enter> para regresar al menú: “);

READ(<Enter>);

FIN

ELSE

INICIO

WRITE(“No existe el archivo. Presiona <Enter> para regresar al menú: “);

READ(<Enter>);

FIN

FIN

Ordenar( Totalelementos|Inicio)

INICIO

Arreglo=(tSimbolo)malloc(sizeof(tSimbolo)\*elementos2);

temp=\*Inicio;

DESDE(i=0)

INICIO

Arreglo[i].Caracter=temp->Caracter;

Arreglo[i].Probabilidad=temp->Probabilidad;

i++;

FIN

HASTA(i<Totalelementos)

DESDE(i=0)

INICIO

DESDE(j=0)

INICIO

IF(Arreglo[i]>Arreglo[j])THEN

INICIO

Hoja=Arreglo[i];

Arreglo[i]=Arreglo[j];

Arreglo[j]=Hoja;

FIN

FIN

HASTA(j<Totalelementos)

FIN

HASTA(i<Totalelementos)

temp=\*Inicio;

i=0;

WHILE(temp->sig!=NULL)

INICIO

temp->Caracter=Arreglo[i].Caracter;

temp->Probabilidad=Arreglo[i].Probabilidad;

i=i+1;

temp=temp->sig;

FIN

free(Arreglo);

FIN

FUNCIÓN Codificar\_Mensaje (Inicio, Long | )

INICIO

Mensaje, Longitud, i=0, MensajeCod, Temp, ERROR=0;

Write(“Dame el mensaje que deseas codificar: “);

Read( &Mensaje);

Longitud=strlen(Mensaje);

WHILE(i<Longitud)

INICIO

Temp=Inicio;

WHILE(Temp!=NULL)

INICIO

IF(Temp->Caracter==Mensaje[i])THEN

INICIO

Concatenar(MensajeCod, Temp->Codigo);

ERROR=1;

FIN

Temp=Temp->Next;

FIN

i=i+1;

IF(ERROR==0)

INICIO

Write(“Error Total: No se cuenta con la suficiente información para codificar. Presiona <enter> para regresar al menú: “);

Read(Enter);

return 1;

FIN

ERROR=0;

FIN

FIN

FUNCION Generar\_Codigo(Inicio, Raiz, Temp)

INICIO

IF(Temp!=NULL)

INICIO

IF(Temp->Caracter!=0)

INICIO

Variable=Inicio;

WHILE(Variable->Caracter!=Temp->Caracter)

INICIO

Variable=Variable->Next;

FIN

COPY(Variable->Codigo, Lista.Codigo):

FIN

COPY(Lista.Codigo, “1”);

Generar\_Codigo(Inicio, Raiz, Temp->Izq);

Lista.Codigo[strlen(Lista.Codigo)-1]=0;

COPY(Lista.Codigo, “0”);

Generar\_Codigo(Inicio, Raiz, Temp->Der);

Lista.Codigo[strlen(Lista.Codigo)-1]=0;

FIN

FIN

FUNCION Borrar\_Arbol(&Raiz)

INICIO

IF(Raiz!=NULL)

INICIO

Borrar\_Arbol(Raiz->Izq);

Borrar\_Arbol(Raiz->Der);

free(Raiz);

Raiz=NULL;

FREE(Raiz);

FIN

FUNCIÓN Guardar\_Diccionario(Inicio |Totalelementos)

INICIO

Variables: Temp=Inicio, Archivo;

IF((OPEN(“Diccionario.txt”, “wt”))==NULL)

INICIO

Write(“ERROR: El archivo no pudo abrirse.”);

FIN

ELSE

INICIO

WHILE(Temp !=NULL)

INICIO

WRITE(Archivo, Temp->Caracter);

WRITE(Archivo, Temp->Probabilidad);

Temp = Temp ->Next;

Totalelementos++;

FIN

CLOSE(Archivo);

Write( | “El diccionario se ha guardado correctamente.”);

FIN

FIN

FUNCIÓN Leer\_Diccionario( | Inicio, Totalelementos)

INICIO

Variables: temp, temp2, Archivo, Car, Prob;

IF((OPEN(“Diccionario.txt”, “rt”))==NULL)

INICIO

Write(“ERROR: El archivo no pudo abrirse.”);

FIN

ELSE

INICIO

Totalelementos=0;

WHILE(!feof)

INICIO

READ(Archivo,Car);

READ(Archivo, Prob);

Totalelementos=Totalelementos+1;

temp=(tSimbolo\*)malloc(sizeof(tSimbolo));

temp->Caracter=Car;

temp->Probabilidad=Prob;

IF(\*Inicio==NULL)THEN

INICIO

temp->sig=NULL;

\*Inicio=temp;

FIN

ELSE

INICIO

temp2=\*Inicio;

WHILE(temp2->sig!=NULL)

INICIO

temp2=temp2->sig;

FIN

temp2->sig=temp;

temp->sig=NULL;

FIN

FIN

FIN

CLOSE(Archivo);

FIN

FIN

FUNCION Borrar\_Lista(|Inicio)

INICIO

Variables: Temp;

WHILE(Inicio!=NULL)

INICIO

Temp=Inicio;

Inicio=Inicio->Next;

free(Temp);

FIN

FIN

Generar\_Arbol(Totalelementos, Inicio | Raiz)

INICIO

elementos2=Totalelementos;

Inicio2=NULL;

tempa=Inicio;

tempc=NULL;

WHILE(temp1!=NULL)

INICIO

tempb=tempc;

tempb=(tSimbolo\*)malloc(sizeof(tSimbolo));

tempb=tempa;

IF(Inicio2==NULL)

INICIO

Inicio2=tempb;

FIN

tempc=tempb->sig;

tempa=tempa->sig;

FIN

temp1=Inicio2;

temp2=temp1->sig;

a.Probabilidad=temp1->Probabilidad;

b.Probabilidad=temp2->Probabilidad;

a.Caracter=temp1->Caracter;

b.Caracter=temp2->Caracter;

c.Caracter=0; //Caracter Nulo

c.Probabilidad=a.Probabilidad+b.Probabilidad;

Temp3=(tArbol\*)malloc(sizeof(tArbol));

Temp3->Caracter=c.Caracter;

Temp3->Probabilidad=c.Probabilidad;

temp11=(tArbol\*)malloc(sizeof(tArbol));

temp22=(tArbol\*)malloc(sizeof(tArbol));

temp11->Probabilidad=a.Probabilidad;

temp22->Probabilidad=b.Probabilidad;

temp11->Caracter=a.Caracter;

temp22->Caracter=b.Caracter;

temp11->izq=NULL;

temp11->der=NULL;

temp22->izq=NULL;

temp22->der=NULL;

Temp3->izq=temp11;

Temp3->der=temp22;

Eliminar2(&Inicio);

Insertar2(c, &Inicio2);

elementos2=elementos2-1;

Ordenar2( elementos2, &Inicio2);

WHILE(Temp3->Probabilidad!=100)

INICIO

temp1=Inicio2;

temp2=Inicio2->sig;

IF(temp1->Probabilidad!=c.Probabilidad)THEN

INICIO

temp11=(tArbol\*)malloc(sizeof(tArbol));

a.Probabilidad=temp1->Probabilidad;

a.Caracter=temp1->Caracter;

temp11->Caracter=a.Caracter;

temp11->Probabilidad=a.Probabilidad;

FIN

ELSE

INICIO

temp11=temp3;

FIN

IF(temp2->Probabilidad!=c.Probabilidad)THEN

INICIO

temp22=(tArbol\*)malloc(sizeof(tArbol));

b.Probabilidad=temp2->Probabilidad;

b.Caracter=temp2->Caracter;

temp22->Caracter=b.Caracter;

temp22->Probabilidad=b.Probabilidad;

FIN

ELSE

INICIO

temp22=temp3;

FIN

c.Caracter=0;

c.Probabilidad=(temp22->Probabilidad)+(temp11->Probabilidad);

Temp3=(tArbol\*)malloc(sizeof(tArbol));

Temp3->Caracter=c.Caracter;

Temp3->Probabilidad=c.Probabilidad;

Temp3->izq=Temp11;

Temp3->der=Temp22;

IF(Temp11->Caracter!=0)THEN

INICIO

Temp11->izq=NULL;

Temp11->der=NULL;

FIN

IF(Temp22->Caracter!=0)THEN

INICIO

Temp22->izq=NULL;

Temp22->der=NULL;

FIN

Eliminar2(&Inicio);

Insertar2(c, &Inicio2);

elementos2=elementos2-1;

Ordenar2( elementos2, &Inicio2);

FIN

FIN

Eliminar2(| Inicio2)

INICIO

temp=\*Inicio2;

temp2=temp->sig;

\*Inicio2=temp2->sig;

free(temp);

free(temp2);

FIN

Insertar2(c |Inicio2)

INICIO

temp=\*Inicio2;

WHILE(temp->sig!=NULL)

INICIO

temp=temp->sig;

FIN

temp2=(tSimbolo\*)malloc(sizeof(tSimbolo));

temp2->Caracter=c.Caracter;

temp2->Probabilidad=c.Probabilidad;

temp->sig=temp2;

temp2->sig=NULL;

FIN

Ordenar2(elementos2 |Inicio2)

INICIO

Arreglo=(tSimbolo)malloc(sizeof(tSimbolo)\*elementos2);

temp=\*Inicio2;

DESDE(i=0)

INICIO

Arreglo[i].Caracter=temp->Caracter;

Arreglo[i].Probabilidad=temp->Probabilidad;

i++;

FIN

HASTA(i<elementos2)

DESDE(i=0)

INICIO

DESDE(j=0)

INICIO

IF(Arreglo[i]>Arreglo[j])THEN

INICIO

Hoja=Arreglo[i];

Arreglo[i]=Arreglo[j];

Arreglo[j]=Hoja;

FIN

FIN

HASTA(j<elementos2)

FIN

HASTA(i<elementos2)

temp=\*Inicio2;

i=0;

WHILE(temp->sig!=NULL)

INICIO

temp->Caracter=Arreglo[i].Caracter;

temp->Probabilidad=Arreglo[i].Probabilidad;

i=i+1;

temp=temp->sig;

FIN

free(Arreglo);

FIN