Estructuras de datos: Cadenas

Cadenas: array de carácteres
Listas de cadenas

1. Introducción.

En sus inicios, los microprocesadores y los programas de computadora se desarrollaron para resolver problemas de cálculo numérico. Sin embargo, a la actualidad esto ha cambiado, pues es conocido que la empresa y en general los procesos comerciales en general gestionan volúmenes de información de tipo cadena o alfanuméricos como direcciones, nóminas de empleados, listas de productos y sus transacciones de ventas para clientes. Hoy en día es cotidiano usar editores de texto, base de datos, etc. En donde las cadenas de caracteres son datos predominantes.

Un **String** o cadena de caracteres, es una secuencia de símbolos que incluyen letras del alfabeto, dígitos, y caracteres especiales.

Tecnologicamenete estos caracteres pertenecen al sistema de codificación ASCII (1 byte) & UNICODE (2 bytes).

Carácter.
Es un tipo primitivo de dato o ***Char***, definido como un símbolo del juego de caracteres de codificación ASCII-UNICODE. Delimitados por comillas simples.

Cadena.
Es una secuencia finita de caracteres o ***string***. y una constante de cadena se representa por un juego de caracteres ASCII-UNICODE, delimitados por comillas dobles.

2. Juego de caracteres para computadora.

Los juegos de caracteres, has sido estandarizados en sistemas de codificación entre las que destacan las Tablas ASCII, EBCDIC y UNICODE. Las computadoras compatibles utilizan los sistemas de Codificación ASCII y UNICODE.

El código ASCII. (American Standard Code for Informaton Interchange)

El código ASCII standard utiliza 7 bits (dígitos binarios: 0, 1) para representar un total de 2^7 (127) caracteres distintos. El código ASCII enhance o ampliado, utiliza 8 bits para representar 2^8 (256) caracteres distintos. Este último es el standard de la PC compatibles.

El código UNICODE (Consorcio Unicode: www.unicode.org).

Es el código universal para internet y alfabetos internacionales. Aunque ASCII es el standard de codificación creado originalmente para el idioma ingles, este es muy limitado para otros idiomas pues es un código de 1 byte (8 bits) que puede representar 256 caracteres diferentes (2^8 =256). El lenguaje Java comenzó a utilizar la representación internacional Unicode mas moderna y mas amplia en juego de caracteres, ya que es un código de 2 bytes (16 bits), que permite representar hasta 65.536 caracteres diferentes (2^{16} =265.536). Con Unicode se resuelve el problema de representación de los alfabetos internacionales, como el Arabe, chino, japonés, etc. e inclusive de idiomas muertos.

Ejemplos Comentados

Problema 1. Algoritmo de búsqueda: Realizar la búsqueda de nombres en una lista de empleados. El resultado de la rutina de búsqueda retorna uno de los siguientes mensajes:

"Nombre encontrado" //si empleado está en la lista
"No se encuentra" //si el empleado no esta en la lista

Análisis:

Implemente un **flag** o ***interruptor*** sw, para señalar sw ← false (no encontrado) o true (encontrado). Un ***bucle de búsqueda*** determinara la presencia del empleado en la nómina. Defina y asigne en un ***array de tipo cadena***, los nombres de la siguiente nómina de empleados:

"Alexandra"
"Fernando"
"Rosario"
"Margarita"
"Rodrigo"
"Nicolás"
"Raquel"
"Daniel"

Pseudocodigo

```
Algoritmo Búsqueda de empleado.
const N = 8        //8 empleados en la nomina
tipo
      array[1:10] de caracter: cadena
tipo
      array[1:N] de cadena: arrEmpleado
var
      arrEmpleado: emp  // tabla de registro de nombres de empleados
      logico:     sw    // interruptor binario (flag)
      cadena:     nombre
      entero:     i
inicio
      //inicializar nomina empleados
      emp[1] :=  "Alejandra"
      emp[2] :=  "Fernando"
      emp[3] :=  "Rosario"
      emp[4] :=  "Margarita"
      emp[5] :=  "Rodrigo"
      emp[6] :=  "Nicolás"
      emp[7] :=  "Raquel"
      emp[8] :=  "Daniel"

      sw ← falso //inicializar interruptor

      escribir("Ingrese nombre empleado: ")
      leer(nombre)
      //buscar nombre
      desde( i := 1 hasta N )hacer
            si( emp[i] = nombre )entonces
                  sw ← verdadero
            fin_si
      fin_desde

      //imprimir resultados
      si( sw = verdadero )entonces
            escribir("Encontrado")
      si_no
            escribir("No existe ", nombre)
      fin_si
fin
```

Problema 2. Implementar un algoritmo de acceso a aplicaciones de Documentos ASCII (*.txt) mediante usuarios con clave.

Análisis.

- El acceso de usuarios implica la validación del nombre de usuario y clave, ingresados por teclado como cadenas de caracteres.

Restriccion 1: El nombre de usuario es una cadena de caracteres con un máximo de 20 caracteres, y el password admite solo 4 dígitos, cuyos caracteres deben visualizarse con asteriscos (protección contra curiosos).

Restriccion 1: El número de intentos para ingresar al sistema de manera satisfactoria es de 3 intentos, después del cual terminara el programa con un mensaje de salida.

- Si el nombre y clave de usuario es validado satisfactoriamente, entonces el programa debe abrir un archivo de texto con el listado de una agenda telefónica de clientes almacenados en el archivo: Agenda.txt en el directorio de trabajo o predeterminado.
- El archivo `Agenda.txt` debe estar creado previamente y cuyo contenido es por ejemplo:

Cliente	Telefono
Agricola S.A.C.	954 763286
Veterinaria Pet	996 246754
…	

Pseudocodigo

```
Algoritmo. Acceso de usuarios.
tipo
      array[1:20] de caracter: cadena
var
    const USR = "utp" //usuario a validar
    const PWD = 1234  //clave a validar

    cadena: usuario     //maximo 19 caracteres +  '\0' fin de cadena o NULL
    cadena: clave       //4 digitos +  '\0' fin de cadena o NULL
    entero: k = 1
    logico: flag = falso

inicio

      hacer
            //invocar función ingresar Usuario
            flag ← getUser(usuario, USR)
            //invocar procedimiento ingresar clave
            llamar_a getPawd(clave)

            //validar acceso de usuario
            si(entero(clave) = PWD Y flag = verdad)entonces
                  ejecutar_comando("NotePad.exe Agenda.txt");
                  k ← 4
            fin_si

            k ← k + 1
      hasta_que (k >= 3)     //tres intentos

      escribir('Acceso: ', _fecha_sistema(), 'Hasta pronto..!')
fin
```

```
//Definición de prototipos
//rutina 1: getUser()
logico función  getUser(E/S cadena: nomUsr, valorUSER)
var
      entero: i=1
      logico: flag = false

inicio

     escribir("Usuario: ") // solo 3 caracteres
     repetir
             nomUsr[i] ← leerCar()
             i++;
     hasta_que (nomUsr[i-1] = 13) //tecla enter
     usuario[i-1] ← NULL;  //insertar fin de cadena

     si(usuario =  valorUSER)entonces
            flag ← verdad
      si_no
            flag ← falso
      fin_si

     devolver (flag)
fin_funcion



//rutina: getPwd()
procedimiento getPwd(E/S cadena: pwd)
var
      entero: i
inicio
    escribir("Password: ") // solo 4 digitos
    desde( i=1 hasta 4 )hacer
        pwd[i] ← leerCar()
        escribir('*')    //visualizar clave con **** (4 astericos)
    fin_desde
    pwd[5] ← NULL   //insertar fin de cadena
fin_procedimiento
```

2. Codigo C++: Modo Proyecto

```
/*Name: AccesoDeUsuario.cpp
  Copyright:
  Author: J.Medianero.A
  Date: 25/09/08 09:42
  Description: Validar Acceso de usuario
               para visualizar documentos de texto
*/

#include <cstdlib>
#include <iostream>
#include <conio.h>

using namespace std;
```

```
bool getUser(char [], const char []);
void getPwd(char clave[]);

int main(int argc, char* argv[]){
    const char USR[] = {"utp"};
    const int PWD = 1234;  //4 digitos

    char usuario[20];    //maximo 19 caracteres +  '\0' fin de cadena o NULL
    char clave[5];       //4 digitos +  '\0' fin de cadena o NULL
    int k = 1;

    do{
         //ingresar usr
         bool flag = getUser(usuario, USR);
         //ingresar pwd
         getPwd(clave);
         //validar acceso de usuario
         if(atoi(clave) == PWD and flag){
                system("NotePad.exe Agenda.txt");
                k=4;
         }
         system("CLS");
         k += 1;
    }while(k <= 3);    //tres intentos
    printf("Acceso: %s\nHasta pronto..!\n", __DATE__);

    system("PAUSE");
    return EXIT_SUCCESS;
}//fin main



//rutina: getUser()
bool getUser(char nomUsr[], const char valorUSER[]){
int i=0;
bool flag = false;

     printf("\nUsuario: "); // solo 3 caracteres
     do{
             nomUsr[i] = getche(); fflush(stdin);
             i++;
     }while(nomUsr[i-1] != 13);

     nomUsr [i-1]= '\0';      //insertar fin de cadena
     flag =(strcmp(nomUsr, valorUSER)== 0) ? true: false; //op. ternario
     //if(strcmp(nomUsr, USER)== 0){ flag = true; }else{ flag = false; }

     return flag;
}


//rutina: getPwd()
void getPwd(char pwd[]){
    printf("\nPassword: ");         // solo 4 digitos
    for(int i=0; i < 4; i++){
        pwd[i] = getch();fflush(stdin);
        printf("%c", '*');          //visualizar clave con **** (4 astericos)
    }
    pwd[5]='\0';                    //insertar fin de cadena
}
```