UNIVERSIDAD

POLITECNICA

DE

PACHUCA

Actividad Integradora

Algoritmos

Ingeniería en Software

Grupo: 03

Participante:

Chávez Merino Ricardo

Septiembre – Diciembre

10 de diciembre del 2019

INDICE

[MARCO TEORICO 2](#_Toc27068827)

[PLANTEAMIENTO DEL PROBLEMA 5](#_Toc27068828)

[CODIFICACION 6](#_Toc27068829)

[FICHAS BIBLIOGRAFICAS 15](#_Toc27068830)

# MARCO TEORICO

**¿Qué es una Función?**

Una función es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor. Las funciones pueden tomar parámetros que modifiquen su funcionamiento. Las funciones son utilizadas para descomponer grandes problemas en tareas simples y para implementar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código. Cuando una función es invocada se le pasa el control a la misma, una vez que esta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.

**Matrices o Arreglos Multidimensionales**

La forma más simple de una matriz multidimensional es la matriz bidimensional. Una matriz bidimensional es prácticamente, una lista sencilla de matrices unidimensionales. Para declarar una matriz de enteros bidimensional de tamaño [x] [y], se escribiría algo como esto en código de lenguaje C++:

Donde *tipo* puede ser cualquier tipo de datos en C (int, char, long, long long, double, entre otros) y arrayName será un identificador de C válido, o variable. Una matriz bidimensional se puede considerar como una tabla que tendrá [x] número de filas y [y] número de columnas. Una matriz bidimensional a, que contiene tres filas y cuatro columnas.

En este sentido, cada elemento de la matriz a se identifica con un nombre de elemento de la forma a [i] [j], donde ‘a’ es el nombre de la matriz, e ‘i’ y ‘j’ son los índices de los espacios de la matriz de carácter único, o mostrar, cada elemento en ‘a

**Algoritmos de burbuja**

El Ordenación de **burbuja** (Bubble Sort) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado.

**Selección por ordenamiento:**

Este algoritmo mejora ligeramente el [algoritmo de la burbuja](https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja). En el caso de tener que ordenar un vector de enteros, esta mejora no es muy sustancial, pero cuando hay que ordenar un vector de estructuras más complejas, la operación intercambiar () sería más costosa en este caso. Este algoritmo realiza muchas menos operaciones intercambiar () que el de la burbuja, por lo que lo mejora en algo. Si la línea comentada con (!) se sustituyera por intercambiar(lista[i], lista[j]) tendríamos una versión del algoritmo de la burbuja (naturalmente eliminando el orden intercambiar del final).

**Algoritmo de intersección**

Este algoritmo está basado en la definición del "interior" de un polígono, ella misma basada en la noción de [índice](https://es.wikipedia.org/wiki/%C3%8Dndice_(an%C3%A1lisis_complejo)). Considera las regiones de índice par como interiores al polígono: esto es conocido también con el nombre de [regla par-impar](https://es.wikipedia.org/wiki/Regla_par-impar). El algoritmo toma dos listas de polígonos como entrada, cada polígono representado como una lista de cumbres conectadas.

**Ordenamiento rápido**

Ordenamiento rápido. El algoritmo fundamental es el siguiente: Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote. Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores.

# PLANTEAMIENTO DEL PROBLEMA

Se debe crear un algoritmo para un trabajador en un restaurante de la universidad politécnica de Pachuca. Para poder acceder a este sistema debemos ser trabajadores y tener un usuario y contraseña para poder acceder a un menú. Este menú es común en restaurantes, como abrir cuentas en un número limitado de 10 mesas, seleccionar que comida comprar en esa cuenta, modificar cuenta por si alguien quiere comprar después, y por último cerrar cuenta y dar el total a pagar(como extra pregunta si el cliente pago), claro antes de cada decisión seleccionar la mesa en que desea hacer esto.

Por último, el usuario podrá salir del sistema.

Tomar en cuenta el siguiente menú y asignar el precio a cada uno que considere bien:

1.-Cereal con leche

2.-Pan de dulce

3.-Hot Cakes

4.-Hot Dog

5.-Sandwich

7.-Hamburguesa

8.-Enchiladas

9.-Chilaquiles

11.-Huevo al Gusto

12.-Tacos de Guizado(1pz)"

13.-Arroz

14.-Sopa

15.-Frijoles

16.-Menu del dia

7.-Jarra de agua(2lt)

18.-Refresco

19.-Cafe

20.-Atole de Vainilla

# CODIFICACION

#include<iostream>

#include<ctime>

#include<stdlib.h>

using namespace std;

void esperar(double t);

#define SIN\_TIPO string

string validacc(string usuario, string psw, string acceso);

void agragaralimento(int mesanum, string alimentos[], int precios[], int cuenta[10][100], int k, int i, float agregar);

void menualimentos(string alimentos[], int precios[]);

int main() {

string acceso;

int agregar;

string alimentos[21];

int cuenta[10][100];

int i;

int k;

int mesanum;

int opcion1;

int pago;

string pagook;

int precios[21];

string psw;

string pswb;

string repetir1;

float total;

string user;

string usuario;

acceso = "no";

repetir1 = "si";

agregar = 0;

for (i=1;i<=9;i++) {

cuenta[i][0] = 0;

}

do {//se inicia ciclo repetitivo para acceso y bienvenida de usuarios del sistema

//se imprime solo una imagen de presentacion con codigo ASCII generado de la pagina http://patorjk.com/software/taag

cout << " " << endl;

cout << "Ingrese al sistema..." << endl;

cout << " " << endl;

cout << "Usuario: >";

cin >> usuario;

cout << "Password: >";

cin >> psw;

acceso = validacc(usuario,psw,acceso); //se envia al subproceso de validacion de usuario

cout << " " << endl;

if (acceso=="si") {

cout << "Bienvenido al sistema " << usuario << "." << endl;

} else {

cout << "Usuario/Password no validos..." << endl;

}

esperar(3\*1000);

system("cls");

} while (acceso!="si");

do {

system("cls");

cout << "Menu de opciones:" << endl; //menu de opciones con funcion switch-case

cout << " " << endl;

cout << "1.- Abrir una cuenta nueva " << endl;

cout << "2.- Agregar alimento extra en cuenta " << endl;

cout << "3.- Leer cuenta " << endl;

cout << "4.- Cerrar Cuenta" << endl;

cout << "5.- Salir del sistema" << endl;

cout << " " << endl;

cin >> opcion1; //Se lee la opcion del menu

menualimentos(alimentos,precios);

switch (opcion1) {

case 1:

cout << "Abrir cuenta nueva" << endl;

cout << " " << endl;

cout << "Indicar numero de Mesa: >";

cin >> mesanum; // se lee el nuero de mesa que se toma como el nuemero de cuenta

while (mesanum<1 || mesanum>10) {//se valida que sea una mesa entre 1 y 10, si no lo es vuelve a pedir un nuero valido

cout << "Indica un numero de mesa valido 1-9..." << endl;

cin >> mesanum;

}

system("cls");

if (cuenta[mesanum][0]>0) {//se valida si la mesa ya tiene una cuenta abierta, checando el valor de la posicion 0 de la variable de arregle cuenta, si es cero no tiene cuenta abierta

cout << " " << endl;

cout << "Esta Mesa tiene una Cuanta abierta actual..." << endl;

esperar(2\*1000); //retraso de 2 segundo para que se lea el mensaje

} else {

cout << " " << endl;

k = 0;

i = 0;

agragaralimento(mesanum,alimentos,precios,cuenta,k,i,agregar); // si la cuenta no esta abierta se manda al subproceso de agregar alimentos

}

break;

case 2:

system("cls");

cout << "-- Agregar Alimento Extra a Cuenta --" << endl;

cout << " " << endl;

cout << "Numero de Mesa:";

cin >> mesanum;// se solicita el nuero de mesa o cuenta a la que se le agrega alimentos

while (mesanum<1 || mesanum>10) {

cout << "Indica un numero de mesa valido 1-9..." << endl; // se valida que sea una mesa valida entre 1-9

cin >> mesanum;

}

if (cuenta[mesanum][0]>0) {//se valida que la cuenta este abierta, checando que el valor de la posicion cero de la variable cuenta sea mayor a cero

k = cuenta[mesanum][0];

i = 0;

agragaralimento(mesanum,alimentos,precios,cuenta,k,i,agregar); //si la cuenta esta abierta se envia al subproceso de agregar alimentos

} else {

cout << " " << endl;

cout << "La Cuenta de la Mesa - " << mesanum << " - No se encuentra abierta" << endl;//mensaje de error para cuando la cuenta indicada no esta abierta

esperar(2\*1000); //retraso de 2 segundo para que se lea el mensaje

}

break;

case 3:

system("cls");

cout << "-- Lectura de Cuenta --" << endl;

cout << " " << endl;

cout << "Numero de Mesa:"; //se pregunta de que mesa se leera la cuenta

cin >> mesanum;

while (mesanum<1 || mesanum>10) {

cout << "Indica un numero de mesa valido 1-9..." << endl; //se valida que sea un numero de cuenta valido

cin >> mesanum;

}

if (cuenta[mesanum][0]>0) {//se valida que la mesa o cuenta este abierta, checando que la posicion 0 de la variable cuenta sea mayor a cero

cout << " " << endl;

cout << "Orden Actual:" << endl;

cout << " " << endl;

for (i=1;i<=cuenta[mesanum][0];i++) {//se imprime los alimentos que se ordenaron en la cuenta solicitada

cout << " " << alimentos[cuenta[mesanum][i]] << endl;

}

cout << " " << endl;

system("pause");

} else {

cout << " " << endl;

cout << "La Cuenta de la Mesa - " << mesanum << " - No se encuentra abierta" << endl; //ensaje de error por si la cuenta no esta abierta

esperar(2\*1000);//retraso de 2 segundo para que se lea el mensaje

}

break;

case 4:

system("cls");

cout << "-- Cierre y Pago de Cuenta --" << endl;

cout << " " << endl;

cout << "Numero de Mesa:";//se pregunta cual es la mesa o cuenta que se quiere cerrar

cin >> mesanum;

while (mesanum<1 || mesanum>10) {

cout << "Indica un numero de mesa valido 1-9..." << endl;// se valida que la mesa o cuenta sea valida

cin >> mesanum;

}

if (cuenta[mesanum][0]>0) {

cout << " " << endl;

cout << "Ticket de venta Final, Mesa - " << mesanum << " - :" << endl;// se imprime la cuenta total con su costo a pagar

cout << " " << endl;

total = 0;

for (i=1;i<=cuenta[mesanum][0];i++) {

cout << " " << alimentos[cuenta[mesanum][i]] << "$" << precios[cuenta[mesanum][i]] << endl;

total = total+precios[cuenta[mesanum][i]];//se suma el costo de los alimentos

}

cout << " " << endl;

cout << " Total A Pagar: $" << total << endl;//se imprime el total a pagar

cout << " " << endl;

pagook = "no";

cout << "Pago realizado? (1-Si,2-No).";//se valida que el pago se realize

cin >> pago;

if (pago==1) {

cout << " " << endl;

cout << " Cuenta de mesa - " << mesanum << " - CERRADA" << endl;

cuenta[mesanum][0] = 0;//la posicion cero de la variable cuenta se reinicia a cero para indicar que la cuenta esta cerrada

esperar(2\*1000);//retraso de 2 segundo para que se lea el mensaje

}

} else {

cout << " " << endl;

cout << "La Cuenta de la Mesa - " << mesanum << " - No se encuentra abierta" << endl;//ensaje de error por si la cuenta no esta abierta

esperar(2\*1000);//retraso de 2 segundo para que se lea el mensaje

}

break;

case 5:

system("cls");

cout << " " << endl;

cout << "-- Salir del sistema --" << endl;

cout << " " << endl;

repetir1 = "no";

for (i=1;i<=9;i++) {//se valida que las posiciones cero de la variables de arreglo cuenta se encuentre todas en cero, lo que indica que ninguna cuenta o mesa esta abierta

if (cuenta[i][0]!=0) {

cout << "La Cuenta de la Mesa - " << i << " - Aun se encuentra Abierta" << endl;//se imprime las variables que aun no esten en cero

repetir1 = "si";

}

}

if (repetir1=="si") {

cout << " " << endl;

cout << "Todas las cuantas deben estar cerradas antes de cerrar el sistema..." << endl;//se imprime mensaje de error para cuando alguna variable no esta en cero

esperar(2\*1000);

}

break;

}

} while (repetir1!="no");

return 0;

}

string validacc(string usuario, string psw, string acceso) {//subproceso de la validacion de acceso

int i;

string pswb[10];

string user[10];

acceso = "no";

user[1] = "Juan";

user[2] = "Ricardo";

user[3] = "jrchh@yahoo.com";

pswb[1] = "123";

pswb[2] = "456";

pswb[3] = "123";

for (i=1;i<=9;i++) {

if (usuario==user[i]) {

if (psw==pswb[i]) {

acceso = "si";

}

}

}

return acceso;

}

void agragaralimento(int mesanum, string alimentos[], int precios[], int cuenta[][100], int k, int i, float agregar) {//subproceso de agregar alimentos a un cuenta

do {

system("cls");

cout << " Menu de Alimentos: " << endl;

cout << " " << endl;

for (i=1;i<=10;i++) {

cout << alimentos[i] << "$" << precios[i] << " " << alimentos[i+10] << "$" << precios[i+10] << endl;// se imprime la lista de alimento del menu, variable alimentos y precio

}

cout << " " << endl;

if (k>0) {

cout << "La Orden para mesa - " << mesanum << "- Contiene " << k << " alimentos:" << endl;//se imprime en caso de haber, los alimentas actualmente ordenados para la mesa indicada

cout << " " << endl;

for (i=1;i<=k;i++) {

cout << alimentos[cuenta[mesanum][i]] << endl;

}

cout << " " << endl;

}

cout << "Que alimento deseas agregar: (0=Salir)";

cin >> agregar;//se lee el numero de alimento que se quiere solicitar

cout << " " << endl;

if (agregar>=1 && agregar<=20) {

k = k+1;

cuenta[mesanum][k] = agregar;//se almacena el numero de alimento en la posicion siguiente vacia de la variable cuenta

}

} while (agregar>0);

cuenta[mesanum][0] = k;

}

void menualimentos(string alimentos[], int precios[]) {//subproceso de escritura de menu en variables alimentos y precio

alimentos[1] = " 1.-Cereal con leche ";

alimentos[2] = " 2.-Pan de dulce ";

alimentos[3] = " 3.-Hot Cakes ";

alimentos[4] = " 4.-Hot Dog ";

alimentos[5] = " 5.-Sandwich ";

alimentos[6] = " 6.-Torta ";

alimentos[7] = " 7.-Hamburguesa ";

alimentos[8] = " 8.-Enchiladas ";

alimentos[9] = " 9.-Chilaquiles ";

alimentos[10] = "10.-Sincronizada (3pz) ";

alimentos[11] = "11.-Huevo al Gusto ";

alimentos[12] = "12.-Tacos de Guizado(1pz)";

alimentos[13] = "13.-Arroz ";

alimentos[14] = "14.-Sopa ";

alimentos[15] = "15.-Frijoles ";

alimentos[16] = "16.-Menu del dia ";

alimentos[17] = "17.-Jarra de agua(2lt) ";

alimentos[18] = "18.-Refresco ";

alimentos[19] = "19.-Cafe ";

alimentos[20] = "20.-Atole de Vainilla ";

precios[1] = 9;

precios[2] = 5;

precios[3] = 15;

precios[4] = 8;

precios[5] = 15;

precios[6] = 15;

precios[7] = 25;

precios[8] = 35;

precios[9] = 35;

precios[10] = 15;

precios[11] = 30;

precios[12] = 15;

precios[13] = 10;

precios[14] = 10;

precios[15] = 10;

precios[16] = 35;

precios[17] = 15;

precios[18] = 15;

precios[19] = 8;

precios[20] = 10;

}

void esperar(double t) {//funcion que permite gererar un retraso de x minutos para leer mensajes teporales

clock\_t t0 = clock();

double e = 0;

do {

e = 1000\*double(clock()-t0)/CLOCKS\_PER\_SEC;

} while (e<t);

}

# FICHAS BIBLIOGRAFICAS

Programación en C++/Funciones. (2017, 13 mayo). Consultado el 10 diciembre, 2019, de <https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B/Funciones>

EcuRed. (2017, 5 mayo). Ordenamiento de burbuja. Consultado el 10 diciembre, 2019, de <https://www.ecured.cu/Ordenamiento_de_burbuja>

Redacción ConceptoDefinicion. (2019, 9 septiembre). Algoritmo Burbuja. Consultado 10 diciembre, 2019, de <https://conceptodefinicion.de/algoritmo/>