**UNIVERSIDAD DE LAS FUERZAS ARMADAS “ESPE”**

**INTEGRANTES:** David Cantuña, Mathius Moyano

**FECHA DE ENTREGA:** 10/06/2022

**DOCENTE:** Ing. Fernando Solís

**ASIGNATURA:** Estructura de datos **NRC:** 4683

**PROYECTO REGISTRO DE VEHICULOS EN UNA MECANICA**

1. **Objetivo**

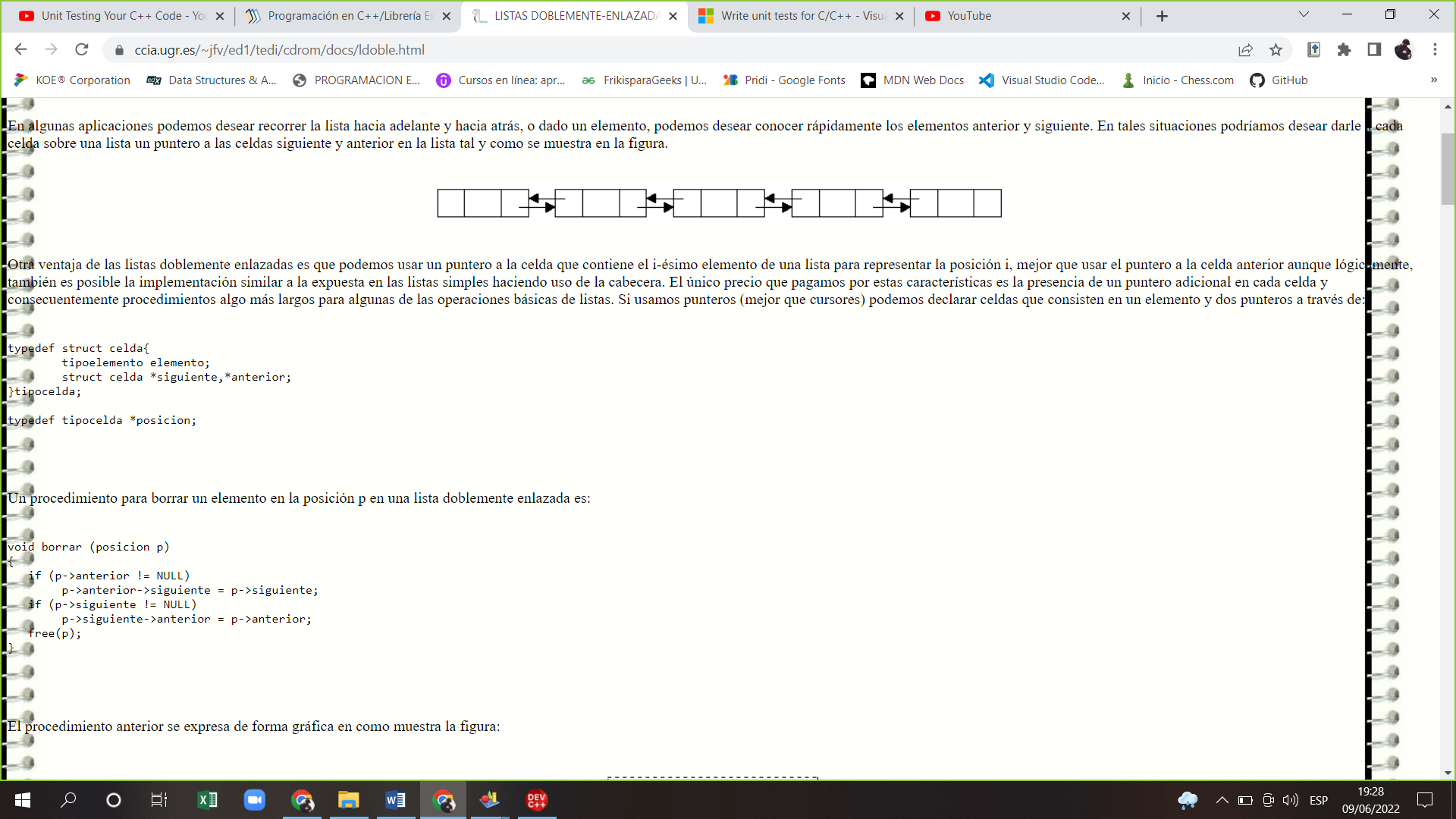
Implementar de manera correcta Estructura de Datos combinado en la práctica con POO.

1. **Introducción**

**Listas Doblemente Enlazadas Circulares.**

Una lista doblemente enlazada es una estructura de datos en la que cada nodo almacena punteros tanto al siguiente como al anterior.

Otra ventaja de las listas doblemente enlazadas es que podemos usar un puntero a la celda que contiene el i-ésimo elemento de una lista para representar la posición i, mejor que usar el puntero a la celda anterior, aunque lógicamente, también es posible la implementación similar a la expuesta en las listas simples haciendo uso de la cabecera. El único precio que pagamos por estas características es la presencia de un puntero adicional en cada celda y consecuentemente procedimientos algo más largos para algunas de las operaciones básicas de listas.



1. **Instrucciones**
   1. **Planteamiento del problema**

Se requiere ingresar los datos de cada vehículo y registrarlos, esto será manejado por el usuario desde un menú donde se puede Ingresar, Buscar, Eliminar y Mostrar.

* 1. **Requerimientos**
* Listas Doblemente Enlazadas Circulares.
* Generar un menú Insertar, Buscar, Generar.
* Ingreso de datos de cada auto.
* Usar POO.
* Buscar por auto y por placa.
  1. **Extras**
* Login
* Almacenar en archivos .txt
* Reporte en pdf
* Foto pixeleada grupo
* Instalador del proyecto
* C++ doc.
* Marquesina
* Ayuda
* Menú opciones con cursores.
  1. **Evaluación.**
* Diagrama clases.
* Codificación.
* Pruebas Unitarias.
* C++ doc.
* WikiLeaks.

1. **Entregables**
   1. **Principal. Cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88 | #include <iostream>  #include <string>  #include "ListaDobleCircular.h"  #include "Menu.h"  #include "Imagen.h"  #include <windows.h>  #include <iostream>  using namespace std;  int main(int argc, char\*\* argv) {  ListaDobleCircular ldc;  Menu menu;  Imagen img;    int opcion;    const char UsuarioC[] = "admin";  const char ContraC[] = "admin";    char usuario[30];  char contrasena[30];      do{  cin.clear();  fflush(stdin);  cout << "\n\n INGRESE EL USUARIO: ";  //menu.Marquesina("INGRESE EL USUARIO: ");  cin.getline(usuario,30);  cout << endl;  cout << "\n INGRESE LA CONTRASENIA: ";  //menu.Marquesina("INGRESE LA CONTRASENIA: ");  cin.getline(contrasena,30);  }while(!((strcmp(UsuarioC,usuario)==0) && (strcmp(ContraC,contrasena)==0)));    do{  opcion = menu.MenuCursor();    switch(opcion){  case 0:  ldc.Agregar();  system("pause>nul"); // Pausa  break;  case 1:  ldc.Mostrar();  system("pause>nul"); // Pausa  break;  case 2:  ldc.Buscar();  system("pause>nul"); // Pausa  break;  case 3:  ldc.Borrar();  system("pause>nul"); // Pausa  break;  case 4:  ldc.GenerarArchivo();  system("pause>nul"); // Pausa  break;  case 5:  HWND console = GetConsoleWindow();  HDC dc = GetDC(console);  img.DrwImage("EasyBMPtext.bmp", &dc);  system("pause>nul"); // Pausa  break;  default:  cout << endl;  //menu.Marquesina("EJECUCION TERMINADA");  break;  }  }while(opcion != 6);    return 0;  } |

* 1. **ListaDobleCircular. cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335 | #include "ListaDobleCircular.h"  #include "Validaciones.h"  #include <iostream>  #include <fstream>  #include <string.h>  using namespace std;  Validaciones val;  ListaDobleCircular::ListaDobleCircular(){  primero = NULL;  ultimo = NULL;  }  void ListaDobleCircular::Agregar(){  Nodo \*nuevo = new Nodo();  cout<<"\t\t\t-------------------"<<endl;  cout<<"\t\t\t|REGISTRO DE DATOS|"<<endl;  cout<<"\t\t\t-------------------"<<endl;  cout<<endl;  do{  cin.clear();  fflush(stdin);  cout<<"\t\t|NOMBRE CLIENTE: ";  cin.getline(nuevo->nombreCliente,30);  }while(val.ValidarCadenas(nuevo->nombreCliente)!=0);    do{  cin.clear();  fflush(stdin);  cout<<"\t\t|CEDULA CLIENTE: ";  cin.getline(nuevo->cedulaCliente,30);  }while(val.ValidarCedulaEc(nuevo->cedulaCliente)==false);    do{  cin.clear();  fflush(stdin);  cout<<"\t\t|MARCAR AUTO: ";  cin.getline(nuevo->marcaA,30);  }while(val.ValidarCadenas(nuevo->marcaA)!=0);    do{  cin.clear();  fflush(stdin);  cout<<"\t\t|NUMERO DE PLACA: ";  cin.getline(nuevo->placaA,30);  }while((val.ValidarMatriculaEc(nuevo->placaA))==false);    do{  cin.clear();  fflush(stdin);  cout<<"\t\t|FECHA DE INGRESO: ";  cin.getline(nuevo->fechaIngreso,30);  }while((val.ValidarFecha(nuevo->fechaIngreso))==false);  if(primero == NULL){  primero = nuevo;  ultimo = nuevo;  primero->siguiente = primero;  primero->anterior = ultimo;  }else{  ultimo->siguiente = nuevo;  nuevo->anterior = ultimo;  nuevo->siguiente = primero;  ultimo = nuevo;  primero->anterior = ultimo;  }  }  void ListaDobleCircular::Mostrar(){  Nodo \*actual = new Nodo();  actual = primero;    if(primero == NULL){  cout << "\t\tNO HAY AUTOS REGISTRADOS" << endl;  }else{    cout << "\t\t\t--------------------------" << endl;  cout << "\t\t\t| REGISTROS |" << endl;  cout << "\t\t\t--------------------------" << endl;  do{  cout << endl;  cout<<"\t\t--------------------------------------------"<<endl;  cout<<"\t\tNOMBRE DEL PROPIETARIO: "<< actual -> nombreCliente <<endl;  cout<<"\t\tNUMERO DE CEDULA: "<< actual -> cedulaCliente <<endl;  cout<<"\t\tMARCA DEL AUTO: "<< actual -> marcaA <<endl;  cout<<"\t\tNUMERO DE PLACA: "<< actual -> placaA <<endl;  cout<<"\t\tFECHA DE INGRESO: "<< actual -> fechaIngreso <<endl;  cout<<"\t\t--------------------------------------------"<<endl;  actual = actual -> siguiente;  }while(actual!=primero);    }  }  /\*  void ListaDobleCircular::Mostrar(){  Nodo \*actual = new Nodo();  actual = primero;    ifstream archivo;  string texto;    archivo.open("registro.txt",ios::in);  if(archivo.fail()){  cout << "\t\tNO HAY AUTOS REGISTRADOS" << endl;  }else{  while(!archivo.eof()){  getline(archivo, texto);  cout << texto << endl;  }  archivo.close();  }  }  \*/  void ListaDobleCircular::Buscar(){  Nodo \*recor = new Nodo();  recor = primero;  char opcionBuscar[2];  int convertir;    if(primero==NULL){  cout<<"\t\t NO EXISTEN REGISTROS "<<endl;  }else{  cout << "\t\t\t---------------------------" << endl;  cout << "\t\t\t |BUSQUEDA DE REGISTROS|" << endl;  cout << "\t\t\t---------------------------" << endl;  cout<<"\n\n";    cout<<"\t\t-----------------------------" << endl;  cout<<"\t\tPOR MARCA AUTO.................[1]"<<endl;  cout<<"\t\tPOR PLACA AUTO.................[2]"<<endl;  cout<<"\t\t-----------------------------" << endl;  cout<<"\t\tINGRESE LA OPCION:";  cin>>opcionBuscar;  convertir = atoi(opcionBuscar);    system("cls");  switch(convertir){  case 1:  char nombre[30];  cout << "\t\t\t--------------------------" << endl;  cout << "\t\t\t| REGISTROS |" << endl;  cout << "\t\t\t--------------------------" << endl;  cout<<"\t\tNOMBRE: ";  fflush(stdin);  cin.getline(nombre,30);  fflush(stdin);  do{  if(strcmp(recor->marcaA,nombre)==0){  cout<<endl;  cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout<<"\t\tNOMBRE DEL PROPIETARIO: "<< recor -> nombreCliente <<endl;  cout<<"\t\tNUMERO DE CEDULA: "<< recor -> cedulaCliente <<endl;  cout<<"\t\tMARCA DEL AUTO: "<< recor -> marcaA <<endl;  cout<<"\t\tNUMERO DE PLACA: "<< recor -> placaA <<endl;  cout<<"\t\tFECHA DE INGRESO: "<< recor -> fechaIngreso <<endl;  cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  }  recor = recor -> siguiente;  }while(recor != primero);  break;    case 2:  char apellidos[20];  cout << "\t\t\t--------------------------" << endl;  cout << "\t\t\t| REGISTROS |" << endl;  cout << "\t\t\t--------------------------" << endl;  cout<<"\t\tAPELLIDO: ";  fflush(stdin);  cin.getline(apellidos,20);  fflush(stdin);  do{  if(strcmp(recor->placaA,apellidos)==0){  cout<<endl;  cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  cout<<"\t\tNOMBRE DEL PROPIETARIO: "<< recor -> nombreCliente <<endl;  cout<<"\t\tNUMERO DE CEDULA: "<< recor -> cedulaCliente <<endl;  cout<<"\t\tMARCA DEL AUTO: "<< recor -> marcaA <<endl;  cout<<"\t\tNUMERO DE PLACA: "<< recor -> placaA <<endl;  cout<<"\t\tFECHA DE INGRESO: "<< recor -> fechaIngreso <<endl;  cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;  }  recor = recor -> siguiente;  }while(recor != primero);  break;  }  }  }  void ListaDobleCircular::Borrar(){  Nodo\* actual = new Nodo();  actual = primero;  Nodo\* antr = new Nodo();  antr = NULL;    bool encontrado = false;    char nodoBuscado[20];  fflush(stdin);  cout << "INGRESE EL NOMBRE DEL CLIENTE: ";  cin.getline(nodoBuscado,20);  fflush(stdin);  if(primero!=NULL){  do{  if(strcmp(actual->nombreCliente,nodoBuscado)==0){  cout << "\n EL CLIENTE ( " << nodoBuscado << " ) HA SIDO ENCONTRADO";  if(actual==primero){  primero = primero->siguiente;  primero->anterior = ultimo;  ultimo->siguiente = primero;  }else if(actual==ultimo){  ultimo = antr;  ultimo->siguiente = primero;  primero->anterior = ultimo;  }else{  antr->siguiente = actual->siguiente;  actual->siguiente->anterior = antr;  }  cout << "\n EL REGISTRO SE ELIMINO CON EXITO \n\n";  encontrado = true;  }  antr = actual;  actual = actual->siguiente;  }while(actual!=primero && encontrado != true);    if(!encontrado){  cout << "\n CLIENTE NO ENCONTRADO \n\n";  }  }else{  cout << "\n LA LISTA SE ENCUENTRA VACIA \n\n";  }  }  void ListaDobleCircular::GenerarArchivo(){  Nodo \*actual = new Nodo();  actual = primero;    ofstream archivo;  archivo.open("registro.txt",ios::out);    if(archivo.fail()){  cout << "NO SE PUDO GENERAR EL ARCHIVO";  return ;  }  if(primero == NULL){  cout << "\t\tNO HAY AUTOS REGISTRADOS" << endl;  }else{  archivo << "\t\t\t--------------------------" << endl;  archivo << "\t\t\t| REGISTROS |" << endl;  archivo << "\t\t\t--------------------------" << endl;  do{  archivo << endl;  archivo<<"\t\t--------------------------------------------"<<endl;  archivo<<"\t\tNOMBRE DEL PROPIETARIO: "<< actual -> nombreCliente <<endl;  archivo<<"\t\tNUMERO DE CEDULA: "<< actual -> cedulaCliente <<endl;  archivo<<"\t\tMARCA DEL AUTO: "<< actual -> marcaA <<endl;  archivo<<"\t\tNUMERO DE PLACA: "<< actual -> placaA <<endl;  archivo<<"\t\tFECHA DE INGRESO: "<< actual -> fechaIngreso <<endl;  archivo<<"\t\t--------------------------------------------"<<endl;  actual = actual -> siguiente;  }while(actual!=primero);  }  }  /\*  void ListaDobleCircular::GenerarArchivo(){  Nodo \*actual = new Nodo();  actual = primero;    ofstream archivo;  archivo.open("registro.txt",ios::out);    if(archivo.fail()){  cout << "NO SE PUDO GENERAR EL ARCHIVO";  }else{  archivo << "\t\t\t--------------------------" << endl;  archivo << "\t\t\t| REGISTROS |" << endl;  archivo << "\t\t\t--------------------------" << endl;  do{  archivo << endl;  archivo<<"\t\t--------------------------------------------"<<endl;  archivo<<"\t\tNOMBRE DEL PROPIETARIO: "<< actual -> nombreCliente <<endl;  archivo<<"\t\tNUMERO DE CEDULA: "<< actual -> cedulaCliente <<endl;  archivo<<"\t\tMARCA DEL AUTO: "<< actual -> marcaA <<endl;  archivo<<"\t\tNUMERO DE PLACA: "<< actual -> placaA <<endl;  archivo<<"\t\tFECHA DE INGRESO: "<< actual -> fechaIngreso <<endl;  archivo<<"\t\t--------------------------------------------"<<endl;  actual = actual -> siguiente;  }while(actual!=primero);  }  }  \*/ |

* 1. **Menu. cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  115 | #include "Menu.h"  #include <iostream>  #include <windows.h>  #include <conio.h>  #include <stdlib.h>  #include <string>  using namespace std;  Menu::Menu(){  \*(menuHead + 0) = " REGISTRAR UN NUEVO AUTO ";//0  \*(menuHead + 1) = " MOSTRAR REGISTROS ";//1  \*(menuHead + 2) = " BUSCAR REGISTRO ";//2  \*(menuHead + 3) = " ELIMINAR REGISTRO ";//3  \*(menuHead + 4) = " GENERAR REPORTE EN PDF ";//4  \*(menuHead + 5) = " FOTO PIXELEADA" ;//5  \*(menuHead + 6) = " SALIR"; //6  }    void Menu::Gotoxy(int x, int y){  HANDLE hcon = GetStdHandle(STD\_OUTPUT\_HANDLE);  COORD dwPos;  dwPos.X = x;  dwPos.Y= y;  SetConsoleCursorPosition(hcon,dwPos);  }  /\*  void Menu::usleep(\_\_int64 usec) {  HANDLE timer;  LARGE\_INTEGER ft;  ft.QuadPart = -(10 \* usec); // Convert to 100 nanosecond interval, negative value indicates relative time  timer = CreateWaitableTimer(NULL, TRUE, NULL);  SetWaitableTimer(timer, &ft, 0, NULL, NULL, 0);  WaitForSingleObject(timer, INFINITE);  CloseHandle(timer);  }  \*/  /\*  void Menu::Marquesina(std::string texto){  unsigned short derecha, izquierda, repite;  int longitud = texto.length();  char banner[100];  strcpy\_s(,,,);  for (repite=1; repite<=1; repite++)  {  for (izquierda=longitud; izquierda>=1; izquierda--)  {  system("cls");  printf("\b");  Menu::Gotoxy(izquierda,1);  printf("%s",banner);  usleep(10000);  }  }    }  \*/  int Menu::MenuCursor(){  int cursor = 0;  char tecla;  for (;;) {  system("cls");  //SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 60);  cout << endl;  cout << endl;  cout << " MENU DE OPCIONES " << endl;  for (int i = 0; i < 7; i++) {  if (cursor == i) {  SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 240);  cout << \*(menuHead + i) << endl;  SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 15);  }  else {  SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 15);  cout << \*(menuHead + i) << endl;  }  }  for (;;) {  tecla = \_getch();  if (tecla == 80) {  cursor++;  if (cursor == 7) {  cursor = 0;  }  break;  }  if (tecla == 72) {  cursor--;  if (cursor == -1) {  cursor = 7;  }  break;  }  if (tecla == 13) {  return cursor;  }  }  }  } |

* 1. **Validaciones. cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88 | #include <iostream>  #include <regex>  #include "Validaciones.h"  using namespace std;  int Validaciones::ValidarCadenas(char cadena[30]){  int i=0, validandoL=0, j;  j = strlen(cadena); //se asigna el numero de caracteres que tiene la cadena a J    while(i < j && validandoL == 0){    if(isalpha(cadena[i])!=0 || cadena[i]==32) // 32 para que acepte espacios tambien  {  i++; //si es una letra avanza de lo contrario entra al else para salirse del ciclo  }else{  validandoL=1;  }  }  return validandoL;  }  bool Validaciones::ValidarMatriculaEc(char matricula[30]){  const regex expReg("[A-Z]{3}-\\d{4}");  return regex\_match(matricula,expReg);  }  bool Validaciones::ValidarCedulaEc(char cedula[30]){  int v[10], num, limite;  limite = strlen(cedula);  if (limite != 10){  return false;  }  if (limite == 10){  for (int i=0;i<10;i++){  num = cedula[i]-48;  v[i] = num;  }  int vec[10], suma, ultimo, ultimon;  if (v[0] == 1 || v[0] == 2){  if (v[2] > 0 && v[2] < 6 ){  for (int i=0; i<9; i+=2){  vec[i] = v[i] \*2;  if (vec[i] >= 10){  vec[i] -= 9;  }  }  for (int i=1; i<9; i+=2){  vec[i] = v[i];  }  suma= 0;  for(int i=0; i<10; i++){  suma = suma + vec[i];  }  if (suma == 10)  ultimo = 0;  else if(suma >10 && suma<20)  ultimo = 20;  else if (suma >20 && suma<30)  ultimo = 30;  else if (suma >30 && suma<40)  ultimo = 40;  else return false;    ultimon = ultimo - suma;    if (ultimon == v[9])  return true;  else  return false;  }  else  return false;  }  }  return true;  }  bool Validaciones::ValidarFecha(char fecha[30]){  const regex expRegF("(0?[1-9]|[12][0-9]|3[01])[\\/](0?[1-9]|1[012])[\\/]([012][0-9])\\d{2}");  return regex\_match(fecha,expRegF);  } |

* 1. **Imagen. cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #include "Imagen.h"  #include <windows.h>  #include <iostream>  #include "EasyBMP.h"  void Imagen::DrwImage(std::string imagePath, HDC\* console){  system("cls");  BMP image;  image.ReadFromFile(imagePath.c\_str());  //image.SetBitDepth(32);  for (int x = 0; x < image.TellWidth(); x++)  {  for (int y = 0; y < image.TellHeight(); y++)  {  int RED = image.GetPixel(x, y).Red;  int GREEN = image.GetPixel(x, y).Green;  int BLUE = image.GetPixel(x, y).Blue;  int ALPHA = image.GetPixel(x, y).Alpha;  COLORREF COLOUR = RGB(RED, GREEN, BLUE);  if (ALPHA == 0) {  SetPixel(\*console, x, y, COLOUR);  }  }  }  } |

* 1. **EasyBMP.cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* EasyBMP Cross-Platform Windows Bitmap Library \*  \* \*  \* Author: Paul Macklin \*  \* email: macklin01@users.sourceforge.net \*  \* support: http://easybmp.sourceforge.net \*  \* \*  \* file: EasyBMP.cpp \*  \* date added: 03-31-2006 \*  \* date modified: 12-01-2006 \*  \* version: 1.06 \*  \* \*  \* License: BSD (revised/modified) \*  \* Copyright: 2005-6 by the EasyBMP Project \*  \* \*  \* description: Actual source file \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

* 1. **Nodo. h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #include <iostream>  #include <string>  using namespace std;  class Nodo{  private:  char nombreCliente[30];  char cedulaCliente[30];  char fechaIngreso[30];  char marcaA[30];  char placaA[30];  Nodo \*siguiente;  Nodo \*anterior;  friend class ListaDobleCircular;  }; |

* 1. **ListaDobleCircular. h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #include <iostream>  #include <string>  #include "Nodo.h"  using namespace std;  class ListaDobleCircular{  private:  Nodo \*primero;  Nodo \*ultimo;    public:  ListaDobleCircular();  void Agregar();  void Mostrar();  void Borrar();  void Buscar();  void GenerarArchivo();  }; |

* 1. **Menu. h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include <iostream>  #include <windows.h>  #include <conio.h>  #include <stdlib.h>  #include <string>  using namespace std;  class Menu{  private:  string \*menuHead = new string[7];  public:  Menu();  int MenuCursor();  void Gotoxy(int,int);  void Marquesina(std::string texto);  void usleep(\_\_int64);  }; |

* 1. **Validaciones. h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class Validaciones{  public:  int ValidarCadenas(char cadena[30]);  bool ValidarMatriculaEc(char cadena[30]);  bool ValidarCedulaEc(char cedula[30]);  bool ValidarFecha(char cedula[30]);  }; |

* 1. **Imagen. h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #include <windows.h>  #include <iostream>  #include "EasyBMP.h"  class Imagen {  public:  void DrwImage(std::string imagePath, HDC\* console);  }; |

* 1. **EasyBMP\_DataStructures.h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* EasyBMP Cross-Platform Windows Bitmap Library \*  \* \*  \* Author: Paul Macklin \*  \* email: macklin01@users.sourceforge.net \*  \* support: http://easybmp.sourceforge.net \*  \* \*  \* file: EasyBMP\_DataStructures.h \*  \* date added: 05-02-2005 \*  \* date modified: 12-01-2006 \*  \* version: 1.06 \*  \* \*  \* License: BSD (revised/modified) \*  \* Copyright: 2005-6 by the EasyBMP Project \*  \* \*  \* description: Defines basic data structures for \*  \* the BMP class \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #ifndef \_EasyBMP\_Custom\_Math\_Functions\_  #define \_EasyBMP\_Custom\_Math\_Functions\_  inline double Square( double number )  { return number\*number; }  inline int IntSquare( int number )  { return number\*number; }  #endif  int IntPow( int base, int exponent );  #ifndef \_EasyBMP\_Defined\_WINGDI  #define \_EasyBMP\_Defined\_WINGDI  typedef unsigned char ebmpBYTE;  typedef unsigned short ebmpWORD;  typedef unsigned int ebmpDWORD;  #endif  #ifndef \_EasyBMP\_DataStructures\_h\_  #define \_EasyBMP\_DataStructures\_h\_  inline bool IsBigEndian()  {  short word = 0x0001;  if((\*(char \*)& word) != 0x01 )  { return true; }  return false;  }  inline ebmpWORD FlipWORD( ebmpWORD in )  { return ( (in >> 8) | (in << 8) ); }  inline ebmpDWORD FlipDWORD( ebmpDWORD in )  {  return ( ((in&0xFF000000)>>24) | ((in&0x000000FF)<<24) |  ((in&0x00FF0000)>>8 ) | ((in&0x0000FF00)<<8 ) );  }  // it's easier to use a struct than a class  // because we can read/write all four of the bytes  // at once (as we can count on them being continuous  // in memory  typedef struct RGBApixel {  ebmpBYTE Blue;  ebmpBYTE Green;  ebmpBYTE Red;  ebmpBYTE Alpha;  } RGBApixel;  class BMFH{  public:  ebmpWORD bfType;  ebmpDWORD bfSize;  ebmpWORD bfReserved1;  ebmpWORD bfReserved2;  ebmpDWORD bfOffBits;  BMFH();  void display( void );  void SwitchEndianess( void );  };  class BMIH{  public:  ebmpDWORD biSize;  ebmpDWORD biWidth;  ebmpDWORD biHeight;  ebmpWORD biPlanes;  ebmpWORD biBitCount;  ebmpDWORD biCompression;  ebmpDWORD biSizeImage;  ebmpDWORD biXPelsPerMeter;  ebmpDWORD biYPelsPerMeter;  ebmpDWORD biClrUsed;  ebmpDWORD biClrImportant;  BMIH();  void display( void );  void SwitchEndianess( void );  };  #endif |

* 1. **EasyBMP\_VariousBMPutilities.h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* EasyBMP Cross-Platform Windows Bitmap Library \*  \* \*  \* Author: Paul Macklin \*  \* email: macklin01@users.sourceforge.net \*  \* support: http://easybmp.sourceforge.net \*  \* \*  \* file: EasyBMP\_VariousBMPutilities.h \*  \* date added: 05-02-2005 \*  \* date modified: 12-01-2006 \*  \* version: 1.06 \*  \* \*  \* License: BSD (revised/modified) \*  \* Copyright: 2005-6 by the EasyBMP Project \*  \* \*  \* description: Various utilities. \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #ifndef \_EasyBMP\_VariousBMPutilities\_h\_  #define \_EasyBMP\_VariousBMPutilities\_h\_  BMFH GetBMFH( const char\* szFileNameIn );  BMIH GetBMIH( const char\* szFileNameIn );  void DisplayBitmapInfo( const char\* szFileNameIn );  int GetBitmapColorDepth( const char\* szFileNameIn );  void PixelToPixelCopy( BMP& From, int FromX, int FromY,  BMP& To, int ToX, int ToY);  void PixelToPixelCopyTransparent( BMP& From, int FromX, int FromY,  BMP& To, int ToX, int ToY,  RGBApixel& Transparent );  void RangedPixelToPixelCopy( BMP& From, int FromL , int FromR, int FromB, int FromT,  BMP& To, int ToX, int ToY );  void RangedPixelToPixelCopyTransparent(  BMP& From, int FromL , int FromR, int FromB, int FromT,  BMP& To, int ToX, int ToY ,  RGBApixel& Transparent );  bool CreateGrayscaleColorTable( BMP& InputImage );  bool Rescale( BMP& InputImage , char mode, int NewDimension );  #endif |

* 1. **EasyBMP.h**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* EasyBMP Cross-Platform Windows Bitmap Library \*  \* \*  \* Author: Paul Macklin \*  \* email: macklin01@users.sourceforge.net \*  \* support: http://easybmp.sourceforge.net \*  \* \*  \* file: EasyBMP.h \*  \* date added: 01-31-2005 \*  \* date modified: 12-01-2006 \*  \* version: 1.06 \*  \* \*  \* License: BSD (revised/modified) \*  \* Copyright: 2005-6 by the EasyBMP Project \*  \* \*  \* description: Main include file \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #ifdef \_MSC\_VER  // MS Visual Studio gives warnings when using  // fopen. But fopen\_s is not going to work well  // with most compilers, and fopen\_s uses different  // syntax than fopen. (i.e., a macro won't work)  // So, we'lll use this:  #define \_CRT\_SECURE\_NO\_DEPRECATE  #endif  #include <iostream>  #include <cmath>  #include <cctype>  #include <cstring>  #ifndef EasyBMP  #define EasyBMP  #ifdef \_\_BCPLUSPLUS\_\_  // The Borland compiler must use this because something  // is wrong with their cstdio file.  #include <stdio.h>  #else  #include <cstdio>  #endif  #ifdef \_\_GNUC\_\_  // If g++ specific code is ever required, this is  // where it goes.  #endif  #ifdef \_\_INTEL\_COMPILER  // If Intel specific code is ever required, this is  // where it goes.  #endif  #ifndef \_DefaultXPelsPerMeter\_  #define \_DefaultXPelsPerMeter\_  #define DefaultXPelsPerMeter 3780  // set to a default of 96 dpi  #endif  #ifndef \_DefaultYPelsPerMeter\_  #define \_DefaultYPelsPerMeter\_  #define DefaultYPelsPerMeter 3780  // set to a default of 96 dpi  #endif  #include "EasyBMP\_DataStructures.h"  #include "EasyBMP\_BMP.h"  #include "EasyBMP\_VariousBMPutilities.h"  #ifndef \_EasyBMP\_Version\_  #define \_EasyBMP\_Version\_ 1.06  #define \_EasyBMP\_Version\_Integer\_ 106  #define \_EasyBMP\_Version\_String\_ "1.06"  #endif  #ifndef \_EasyBMPwarnings\_  #define \_EasyBMPwarnings\_  #endif  void SetEasyBMPwarningsOff( void );  void SetEasyBMPwarningsOn( void );  bool GetEasyBMPwarningState( void );  #endif |

* 1. **A**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* EasyBMP Cross-Platform Windows Bitmap Library \*  \* \*  \* Author: Paul Macklin \*  \* email: macklin01@users.sourceforge.net \*  \* support: http://easybmp.sourceforge.net \*  \* \*  \* file: EasyBMP\_VariousBMPutilities.h \*  \* date added: 05-02-2005 \*  \* date modified: 12-01-2006 \*  \* version: 1.06 \*  \* \*  \* License: BSD (revised/modified) \*  \* Copyright: 2005-6 by the EasyBMP Project \*  \* \*  \* description: Defines BMP class \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #ifndef \_EasyBMP\_BMP\_h\_  #define \_EasyBMP\_BMP\_h\_  bool SafeFread( char\* buffer, int size, int number, FILE\* fp );  bool EasyBMPcheckDataSize( void );  class BMP  {private:  int BitDepth;  int Width;  int Height;  RGBApixel\*\* Pixels;  RGBApixel\* Colors;  int XPelsPerMeter;  int YPelsPerMeter;  ebmpBYTE\* MetaData1;  int SizeOfMetaData1;  ebmpBYTE\* MetaData2;  int SizeOfMetaData2;    bool Read32bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Read24bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Read8bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Read4bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Read1bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );    bool Write32bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Write24bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Write8bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Write4bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );  bool Write1bitRow( ebmpBYTE\* Buffer, int BufferSize, int Row );    ebmpBYTE FindClosestColor( RGBApixel& input );  public:  int TellBitDepth( void );  int TellWidth( void );  int TellHeight( void );  int TellNumberOfColors( void );  void SetDPI( int HorizontalDPI, int VerticalDPI );  int TellVerticalDPI( void );  int TellHorizontalDPI( void );    BMP();  BMP( BMP& Input );  ~BMP();  RGBApixel\* operator()(int i,int j);    RGBApixel GetPixel( int i, int j ) const;  bool SetPixel( int i, int j, RGBApixel NewPixel );    bool CreateStandardColorTable( void );    bool SetSize( int NewWidth, int NewHeight );  bool SetBitDepth( int NewDepth );  bool WriteToFile( const char\* FileName );  bool ReadFromFile( const char\* FileName );    RGBApixel GetColor( int ColorNumber );  bool SetColor( int ColorNumber, RGBApixel NewColor );  };  #endif |