|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería De Software | **GUÍA**  No. 02 | **TIEMPO ESTIMADO:**  - |
| **ASIGNATURA:**  Estructura de Datos NRC: 2967 | **FECHA DE ELABORACION:** 27-01-2020  **SEMESTRE**: Septiembre 2019 – Febrero 2020 | |
| **TÍTULO:**  Proyecto Parcial 2   * Jorge Galarza * Kevin Zurita | **DOCENTE:** Ing. Fernando Solís. MsC. | |

**OBJETIVO**

* Desarrollar un programa en C++ que contenga todo el contenido aprendido en segundo parcial de la materia Estructura de Datos, aplicando listas, específicamente listas circulares dobles.

**INSTRUCCIONES**

1. Utilice como material principal, aquel indicado en clase por el docente.
2. Utilice información consultada en Internet y conocimiento adquirido en clase.

**ACTIVIDADES**

1. **Ubicación de recursos**

**p.e.**

1. Formar grupos de máximo 2 personas por computador
2. Instalar la herramienta Tiberius
3. **Planteamiento del problema**

**p.e.**

El programa consiste en realizar un juego al estilo tetris con listas circulares dobles, en el que cae un número de un color, se inserta en la lista y se muestra en pantalla, si vuelve a caer el mismo número del mismo color encima o a un lado del número anterior, se insertará a la lista y se eliminará; caso contrario se añadirá a la lista y esperará a que caiga un numero semejante para volverse a eliminar. Si se llena las filas pierde el juego.

**Marco Teórico**

**Listas:** Listas (Informática). Una lista es una colección de elementos homogéneos entre los que existe una relación lineal.

1. Cada elemento de la lista, a excepción del primero, tiene un único predecesor.

2. Cada elemento de la lista, a excepción del último, tiene un único sucesor.

**Listas Circulares Dobles:** En una lista enlazada doblemente circular, cada nodo tiene dos enlaces, similares a los de la lista doblemente enlazada, excepto que el enlace anterior del primer nodo apunta al último y el enlace siguiente del último nodo, apunta al primero. Como en una lista doblemente enlazada, las inserciones y eliminaciones pueden ser hechas desde cualquier punto con acceso a algún nodo cercano. Aunque estructuralmente una lista circular doblemente enlazada no tiene ni principio ni fin, un puntero de acceso externo puede establecer el nodo apuntado que está en la cabeza o al nodo cola, y así mantener el orden tan bien como en una lista doblemente enlazada.

**Tetris:** fue creado el 6 de junio de 1985 por el inventor Alexei Pajitnov, quien entonces trabajaba en la Academia Soviética de la Ciencia, en Moscú. Su juego, que consistía básicamente en ir encajando piezas de diferentes formas y tamaños que caen desde la parte superior de la pantalla para completar un muro sin dejar huecos, fue lanzado al mercado durante la Perestroika, el plan diseñado por Mijaíl Gorbachov para relanzar la economía soviética. El juego original era en una pantalla en blanco y negro, sin sonido. El campo estaba lleno de asteriscos, y los cuadrados se hacían con dos corchetes enfrentados. Se dice que, aunque fue un éxito rotundo (se vendieron más de 40 millones de copias), Pajitnov no recibió beneficios económicos de su venta; en la actualidad, vive en Estados Unidos y trabaja desarrollando nuevas versiones de su juego para la firma Nintendo..

**Conclusión:**

* Se puede realizar un Tetris aplicando listas circulares dobles y todas sus operaciones en C++ para el correcto funcionamiento del programa, aprovechando la utilización de memoria dinámica.

**Código:**

//CLASE MAIN

/\*\*

\* This file has no copyright assigned and is placed in the Public Domain.

\* This file is part of the mingw-w64 runtime package.

\* No warranty is given; refer to the file DISCLAIMER.PD within this package.

\*/

#ifndef \_TIME\_H\_

#define \_TIME\_H\_

#include <crtdefs.h>

#ifndef \_WIN32

#error Only Win32 target is supported!

#endif

#pragma pack(push,\_CRT\_PACKING)

#ifdef \_\_cplusplus

extern "C" **{**

#endif

#ifndef \_CRTIMP

#define \_CRTIMP \_\_declspec(dllimport)

#endif

#ifndef \_WCHAR\_T\_DEFINED

#define \_WCHAR\_T\_DEFINED

**typedef** unsigned short wchar\_t**;**

#endif

#ifndef \_TIME32\_T\_DEFINED

#define \_TIME32\_T\_DEFINED

**typedef** long \_\_time32\_t**;**

#endif

#ifndef \_TIME64\_T\_DEFINED

#define \_TIME64\_T\_DEFINED

\_\_MINGW\_EXTENSION **typedef** \_\_int64 \_\_time64\_t**;**

#endif

#ifndef \_TIME\_T\_DEFINED

#define \_TIME\_T\_DEFINED

#ifdef \_USE\_32BIT\_TIME\_T

**typedef** \_\_time32\_t time\_t**;**

#else

**typedef** \_\_time64\_t time\_t**;**

#endif

#endif

#ifndef \_CLOCK\_T\_DEFINED

#define \_CLOCK\_T\_DEFINED

**typedef** long clock\_t**;**

#endif

#ifndef \_SIZE\_T\_DEFINED

#define \_SIZE\_T\_DEFINED

#undef size\_t

#ifdef \_WIN64

\_\_MINGW\_EXTENSION **typedef** unsigned \_\_int64 size\_t**;**

#else

**typedef** unsigned int size\_t**;**

#endif

#endif

#ifndef \_SSIZE\_T\_DEFINED

#define \_SSIZE\_T\_DEFINED

#undef ssize\_t

#ifdef \_WIN64

\_\_MINGW\_EXTENSION **typedef** \_\_int64 ssize\_t**;**

#else

**typedef** int ssize\_t**;**

#endif

#endif

#ifndef NULL

#ifdef \_\_cplusplus

#ifndef \_WIN64

#define NULL 0

#else

#define NULL 0LL

#endif /\* W64 \*/

#else

#define NULL ((void \*)0)

#endif

#endif

#ifndef \_TM\_DEFINED

#define \_TM\_DEFINED

struct tm **{**

int tm\_sec**;**

int tm\_min**;**

int tm\_hour**;**

int tm\_mday**;**

int tm\_mon**;**

int tm\_year**;**

int tm\_wday**;**

int tm\_yday**;**

int tm\_isdst**;**

**};**

#endif

#define CLOCKS\_PER\_SEC 1000

\_\_MINGW\_IMPORT int \_daylight**;**

\_\_MINGW\_IMPORT long \_dstbias**;**

\_\_MINGW\_IMPORT long \_timezone**;**

\_\_MINGW\_IMPORT char **\*** \_tzname**[**2**];**

\_CRTIMP errno\_t \_\_cdecl \_get\_daylight**(**int **\***\_Daylight**);**

\_CRTIMP errno\_t \_\_cdecl \_get\_dstbias**(**long **\***\_Daylight\_savings\_bias**);**

\_CRTIMP errno\_t \_\_cdecl \_get\_timezone**(**long **\***\_Timezone**);**

\_CRTIMP errno\_t \_\_cdecl \_get\_tzname**(**size\_t **\***\_ReturnValue**,**char **\***\_Buffer**,**size\_t \_SizeInBytes**,**int \_Index**);**

char **\***\_\_cdecl asctime**(**const struct tm **\***\_Tm**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

char **\***\_\_cdecl \_ctime32**(**const \_\_time32\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

clock\_t \_\_cdecl clock**(**void**);**

double \_\_cdecl \_difftime32**(**\_\_time32\_t \_Time1**,**\_\_time32\_t \_Time2**);**

struct tm **\***\_\_cdecl \_gmtime32**(**const \_\_time32\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

struct tm **\***\_\_cdecl \_localtime32**(**const \_\_time32\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

size\_t \_\_cdecl strftime**(**char **\*** \_\_restrict\_\_ \_Buf**,**size\_t \_SizeInBytes**,**const char **\*** \_\_restrict\_\_ \_Format**,**const struct tm **\*** \_\_restrict\_\_ \_Tm**);**

\_CRTIMP size\_t \_\_cdecl \_strftime\_l**(**char **\*** \_\_restrict\_\_ \_Buf**,**size\_t \_Max\_size**,**const char **\*** \_\_restrict\_\_ \_Format**,**const struct tm **\*** \_\_restrict\_\_ \_Tm**,**\_locale\_t \_Locale**);**

\_CRTIMP char **\***\_\_cdecl \_strdate**(**char **\***\_Buffer**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_CRTIMP char **\***\_\_cdecl \_strtime**(**char **\***\_Buffer**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_\_time32\_t \_\_cdecl \_time32**(**\_\_time32\_t **\***\_Time**);**

\_\_time32\_t \_\_cdecl \_mktime32**(**struct tm **\***\_Tm**);**

\_\_time32\_t \_\_cdecl \_mkgmtime32**(**struct tm **\***\_Tm**);**

#if defined (\_POSIX\_) || defined(\_\_GNUC\_\_)

void \_\_cdecl tzset**(**void**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_MSVC2005**;**

#endif

#if !defined (\_POSIX\_)

\_CRTIMP void \_\_cdecl \_tzset**(**void**);**

#endif

double \_\_cdecl \_difftime64**(**\_\_time64\_t \_Time1**,**\_\_time64\_t \_Time2**);**

\_CRTIMP char **\***\_\_cdecl \_ctime64**(**const \_\_time64\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_CRTIMP struct tm **\***\_\_cdecl \_gmtime64**(**const \_\_time64\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_CRTIMP struct tm **\***\_\_cdecl \_localtime64**(**const \_\_time64\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_CRTIMP \_\_time64\_t \_\_cdecl \_mktime64**(**struct tm **\***\_Tm**);**

\_CRTIMP \_\_time64\_t \_\_cdecl \_mkgmtime64**(**struct tm **\***\_Tm**);**

\_CRTIMP \_\_time64\_t \_\_cdecl \_time64**(**\_\_time64\_t **\***\_Time**);**

unsigned \_\_cdecl \_getsystime**(**struct tm **\***\_Tm**);**

unsigned \_\_cdecl \_setsystime**(**struct tm **\***\_Tm**,**unsigned \_MilliSec**);**

#ifndef \_WTIME\_DEFINED

\_CRTIMP wchar\_t **\***\_\_cdecl \_wasctime**(**const struct tm **\***\_Tm**);**

wchar\_t **\***\_\_cdecl \_wctime32**(**const \_\_time32\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

size\_t \_\_cdecl wcsftime**(**wchar\_t **\*** \_\_restrict\_\_ \_Buf**,**size\_t \_SizeInWords**,**const wchar\_t **\*** \_\_restrict\_\_ \_Format**,**const struct tm **\*** \_\_restrict\_\_ \_Tm**);**

\_CRTIMP size\_t \_\_cdecl \_wcsftime\_l**(**wchar\_t **\*** \_\_restrict\_\_ \_Buf**,**size\_t \_SizeInWords**,**const wchar\_t **\*** \_\_restrict\_\_ \_Format**,**const struct tm **\*** \_\_restrict\_\_ \_Tm**,**\_locale\_t \_Locale**);**

\_CRTIMP wchar\_t **\***\_\_cdecl \_wstrdate**(**wchar\_t **\***\_Buffer**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_CRTIMP wchar\_t **\***\_\_cdecl \_wstrtime**(**wchar\_t **\***\_Buffer**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

\_CRTIMP wchar\_t **\***\_\_cdecl \_wctime64**(**const \_\_time64\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

#if !defined (RC\_INVOKED) && !defined (\_INC\_WTIME\_INL)

#define \_INC\_WTIME\_INL

wchar\_t **\***\_\_cdecl \_wctime**(**const time\_t **\*)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

#ifndef \_\_CRT\_\_NO\_INLINE

#ifndef \_USE\_32BIT\_TIME\_T

\_\_CRT\_INLINE wchar\_t **\***\_\_cdecl \_wctime**(**const time\_t **\***\_Time**)** **{** **return** \_wctime64**(**\_Time**);** **}**

#else

\_\_CRT\_INLINE wchar\_t **\***\_\_cdecl \_wctime**(**const time\_t **\***\_Time**)** **{** **return** \_wctime32**(**\_Time**);** **}**

#endif

#endif /\* \_\_CRT\_\_NO\_INLINE \*/

#endif

#define \_WTIME\_DEFINED

#endif

#ifndef RC\_INVOKED

double \_\_cdecl difftime**(**time\_t \_Time1**,**time\_t \_Time2**);**

char **\***\_\_cdecl ctime**(**const time\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

struct tm **\***\_\_cdecl gmtime**(**const time\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

struct tm **\***\_\_cdecl localtime**(**const time\_t **\***\_Time**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_SEC\_WARN**;**

#ifdef \_POSIX

#ifdef \_\_GNUC\_\_ /\* FIXME: Other compilers that these macros work with? \*/

#ifndef localtime\_r

#define localtime\_r(\_Time, \_Tm) ({ struct tm \*\_\_\_tmp\_tm = \

localtime((\_Time)); \

if (\_\_\_tmp\_tm) { \

\*(\_Tm) = \*\_\_\_tmp\_tm; \

\_\_\_tmp\_tm = (\_Tm); \

} \

\_\_\_tmp\_tm; })

#endif

#ifndef gmtime\_r

#define gmtime\_r(\_Time,\_Tm) ({ struct tm \*\_\_\_tmp\_tm = \

gmtime((\_Time)); \

if (\_\_\_tmp\_tm) { \

\*(\_Tm) = \*\_\_\_tmp\_tm; \

\_\_\_tmp\_tm = (\_Tm); \

} \

\_\_\_tmp\_tm; })

#endif

#ifndef ctime\_r

#define ctime\_r(\_Time,\_Str) ({ char \*\_\_\_tmp\_tm = ctime((\_Time)); \

if (\_\_\_tmp\_tm) \

\_\_\_tmp\_tm = \

strcpy((\_Str),\_\_\_tmp\_tm); \

\_\_\_tmp\_tm; })

#endif

#ifndef asctime\_r

#define asctime\_r(\_Tm, \_Buf) ({ char \*\_\_\_tmp\_tm = asctime((\_Tm)); \

if (\_\_\_tmp\_tm) \

\_\_\_tmp\_tm = \

strcpy((\_Buf),\_\_\_tmp\_tm);\

\_\_\_tmp\_tm; })

#endif

#else /\* NOT GCC: \*/

/\* FIXME: These are more generic but call the main function twice! \*/

#ifndef localtime\_r

#define localtime\_r(\_Time, \_Tm) (localtime ((\_Time)) ? \*(\_Tm) = \*localtime ((\_Time),(\_Tm)) : 0)

#endif

#ifndef gmtime\_r

#define gmtime\_r(\_Time,\_Tm) (gmtime ((\_Time)) ? (\*(\_Tm) = \*gmtime (\_Time),(\_Tm)) : 0)

#endif

#ifndef ctime\_r

#define ctime\_r(\_Time,\_Str) (ctime ((\_Time)) ? (strcpy((\_Str),ctime ((\_Time))),(\_Str)) : 0)

#endif

#endif /\* \_\_GNUC\_\_ \*/

#endif /\* \_POSIX \*/

time\_t \_\_cdecl mktime**(**struct tm **\***\_Tm**);**

time\_t \_\_cdecl \_mkgmtime**(**struct tm **\***\_Tm**);**

time\_t \_\_cdecl time**(**time\_t **\***\_Time**);**

#ifndef \_\_CRT\_\_NO\_INLINE

#if !defined(\_USE\_32BIT\_TIME\_T)

\_\_CRT\_INLINE double \_\_cdecl difftime**(**time\_t \_Time1**,**time\_t \_Time2**)**

**{** **return** \_difftime64**(**\_Time1**,**\_Time2**);** **}**

\_\_CRT\_INLINE char **\***\_\_cdecl ctime**(**const time\_t **\***\_Time**)** **{** **return** \_ctime64**(**\_Time**);** **}**

\_\_CRT\_INLINE struct tm **\***\_\_cdecl gmtime**(**const time\_t **\***\_Time**)** **{** **return** \_gmtime64**(**\_Time**);** **}**

\_\_CRT\_INLINE struct tm **\***\_\_cdecl localtime**(**const time\_t **\***\_Time**)** **{** **return** \_localtime64**(**\_Time**);** **}**

\_\_CRT\_INLINE time\_t \_\_cdecl mktime**(**struct tm **\***\_Tm**)** **{** **return** \_mktime64**(**\_Tm**);** **}**

\_\_CRT\_INLINE time\_t \_\_cdecl \_mkgmtime**(**struct tm **\***\_Tm**)** **{** **return** \_mkgmtime64**(**\_Tm**);** **}**

\_\_CRT\_INLINE time\_t \_\_cdecl time**(**time\_t **\***\_Time**)** **{** **return** \_time64**(**\_Time**);** **}**

#else

\_\_CRT\_INLINE double \_\_cdecl difftime**(**time\_t \_Time1**,**time\_t \_Time2**)**

**{** **return** \_difftime32**(**\_Time1**,**\_Time2**);** **}**

\_\_CRT\_INLINE char **\***\_\_cdecl ctime**(**const time\_t **\***\_Time**)** **{** **return** \_ctime32**(**\_Time**);** **}**

\_\_CRT\_INLINE struct tm **\***\_\_cdecl localtime**(**const time\_t **\***\_Time**)** **{** **return** \_localtime32**(**\_Time**);** **}**

\_\_CRT\_INLINE time\_t \_\_cdecl mktime**(**struct tm **\***\_Tm**)** **{** **return** \_mktime32**(**\_Tm**);** **}**

\_\_CRT\_INLINE struct tm **\***\_\_cdecl gmtime**(**const time\_t **\***\_Time**)** **{** **return** \_gmtime32**(**\_Time**);** **}**

\_\_CRT\_INLINE time\_t \_\_cdecl \_mkgmtime**(**struct tm **\***\_Tm**)** **{** **return** \_mkgmtime32**(**\_Tm**);** **}**

\_\_CRT\_INLINE time\_t \_\_cdecl time**(**time\_t **\***\_Time**)** **{** **return** \_time32**(**\_Time**);** **}**

#endif /\* !\_USE\_32BIT\_TIME\_T \*/

#endif /\* !\_\_CRT\_\_NO\_INLINE \*/

#endif /\* !RC\_INVOKED \*/

#if !defined(NO\_OLDNAMES) || defined(\_POSIX)

#define CLK\_TCK CLOCKS\_PER\_SEC

\_CRTIMP extern int daylight**;**

\_CRTIMP extern long timezone**;**

\_CRTIMP extern char **\***tzname**[**2**];**

void \_\_cdecl tzset**(**void**)** \_\_MINGW\_ATTRIB\_DEPRECATED\_MSVC2005**;**

#endif

#include <\_timeval.h>

#ifndef \_TIMEZONE\_DEFINED /\* also in sys/time.h \*/

#define \_TIMEZONE\_DEFINED

struct timezone **{**

int tz\_minuteswest**;**

int tz\_dsttime**;**

**};**

extern int \_\_cdecl mingw\_gettimeofday **(**struct timeval **\***p**,** struct timezone **\***z**);**

#endif /\* \_TIMEZONE\_DEFINED \*/

#ifdef \_\_cplusplus

**}**

#endif

#pragma pack(pop)

#include <sec\_api/time\_s.h>

/\* Adding timespec definition. \*/

#include <sys/timeb.h>

/\* POSIX 2008 says clock\_gettime and timespec are defined in time.h header,

but other systems - like Linux, Solaris, etc - tend to declare such

recent extensions only if the following guards are met. \*/

#if !defined(IN\_WINPTHREAD) && \

((!defined(\_STRICT\_STDC) && !defined(\_\_XOPEN\_OR\_POSIX)) || \

(\_POSIX\_C\_SOURCE > 2) || defined(\_\_EXTENSIONS\_\_))

#include <pthread\_time.h>

#endif

#endif /\* End \_TIME\_H\_ \*/

//CLASE JUGADOR

#include <iostream>

#include <stdio.h>

class Jugador**{**

private**:**

string nombre\_jugador**;**

int puntaje\_jugador**;**

public**:**

Jugador**(**string**,** int**);**

Jugador**();**

string getNombre**();**

void setNombre**(**string**);**

int getPuntaje**();**

void setPuntaje**(**int**);**

**};**

Jugador**::**Jugador**(**string name**,** int points**){**

nombre\_jugador**=**name**;**

puntaje\_jugador**=**points**;**

**}**

Jugador**::**Jugador**(){**

**}**

string Jugador**::**getNombre**(){**

**return** nombre\_jugador**;**

**}**

void Jugador**::**setNombre**(**string name**){**

nombre\_jugador**=**name**;**

**}**

int Jugador**::**getPuntaje**(){**

**return** puntaje\_jugador**;**

**}**

void Jugador**::**setPuntaje**(**int points**){**

puntaje\_jugador**=**points**;**

**}**

//CLASE LISTACD

#include <iostream>

#include <stdio.h>

#include <string>

#include <fstream>

#include "NodoTetris.h"

**using** **namespace** std**;**

**typedef** NodoTetris**\*** pnodo**;**

class ListaCD**{**

private**:**

pnodo primero**;**

pnodo ultimo**;**

public**:**

ListaCD**();**

void insertarPrimero**(**int**,** int**);**

void insertarUltimo**(**int**,** int**);**

void insertarEntre**(**int**,** int**);**

int chequeoEliminar**(**int**);**

void mostrar**();**

int numeroFila**();**

bool listaVacia**();**

int numeroElementos**();**

void unirLista**();**

pnodo returnNodo**(**int**);**

void generarArchivo**();**

**};**

ListaCD**::**ListaCD**(){**

primero**=NULL;**

ultimo**=NULL;**

**}**

bool ListaCD**::**listaVacia**(){**

**return** primero **==** **NULL;**

**}**

void ListaCD**::**insertarPrimero**(**int value**,** int column**){**

pnodo anterior**;**

pnodo siguiente**;**

**if(**listaVacia**())** **{**

primero **=** **new** NodoTetris**(**value**,**1**,**column**);**

primero**->**sig**=**primero**;**

primero**->**ant**=**primero**;**

ultimo**=**primero**;**

**}** **else** **{**

**if(**ultimo**==**primero**){**

anterior **=** primero**;**

column**=**anterior**->**getColumnas**()-**1**;**

**if(**column**==**0**){**

column**=**1**;**

anterior**->**setColumnas**(**2**);**

**}**

primero **=** **new** NodoTetris**(**value**,**numeroFila**(),**column**);**

primero**->**sig**=**anterior**;**

anterior**->**ant**=**primero**;**

anterior**->**sig**=**primero**;**

ultimo**=**anterior**;**

**}**

**else{**

anterior **=** primero**;**

column**=**anterior**->**getColumnas**()-**1**;**

**if(**column**==**0**){**

column**=**1**;**

**}**

primero **=** **new** NodoTetris**(**value**,**numeroFila**(),**column**);**

primero**->**sig**=**anterior**;**

anterior**->**ant**=**primero**;**

ultimo**->**sig**=**primero**;**

pnodo aux**=**primero**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**ant**->**getColumnas**()+**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**}**

void ListaCD**::**insertarUltimo**(**int value**,** int column**){**

pnodo temp**;**

**if(**listaVacia**())** **{**

primero **=** **new** NodoTetris**(**value**,**1**,**column**);**

primero**->**sig**=**primero**;**

primero**->**ant**=**primero**;**

ultimo**=**primero**;**

**}** **else** **{**

**if(**ultimo**==**primero**){**

temp **=** **new** NodoTetris**(**value**,**numeroFila**(),**primero**->**getColumnas**()+**1**);**

primero**->**sig**=**temp**;**

temp**->**sig**=**primero**;**

temp**->**ant**=**primero**;**

ultimo**=**temp**;**

**}**

**else{**

temp **=** **new** NodoTetris**(**value**,**numeroFila**(),**ultimo**->**getColumnas**()+**1**);**

ultimo**->**sig**=**temp**;**

temp**->**ant**=**ultimo**;**

temp**->**sig**=**primero**;**

ultimo**=**temp**;**

**}**

**}**

**}**

void ListaCD**::**insertarEntre**(**int value**,** int column**){**

pnodo aux**;**

pnodo sig**;**

pnodo control**;**

aux**=**primero**;**

**if(**listaVacia**()){**

insertarPrimero**(**value**,**column**);**

**}else{**

**if(**column**<=**primero**->**getColumnas**()&&**numeroFila**()==**primero**->**getFilas**()){**

insertarPrimero**(**value**,**column**);**

**}**

**else** **if(**column**>=**ultimo**->**getColumnas**()&&**numeroFila**()==**ultimo**->**getFilas**()){**

insertarUltimo**(**value**,**column**);**

**}**

**else{**

**if(**numeroFila**()!=**primero**->**getFilas**()){**

**if(**ultimo**->**getFilas**()!=**numeroFila**()){**

pnodo temp**=new** NodoTetris**(**value**,**numeroFila**(),**column**);**

ultimo**->**sig**=**temp**;**

temp**->**ant**=**ultimo**;**

temp**->**sig**=**primero**;**

ultimo**=**temp**;**

**}else{**

**while(**aux**->**getFilas**()!=**numeroFila**()){**

aux**=**aux**->**sig**;**

**}**

control**=**aux**;**

**if(**column**<=**aux**->**getColumnas**()){**

int x**=** aux**->**getColumnas**()-**1**;**

**if(**x**<=**0**){**

x**=**1**;**

**}**

pnodo temp**=new** NodoTetris**(**value**,**numeroFila**(),**x**);**

sig**=**aux**->**ant**;**

sig**->**sig**=**temp**;**

temp**->**ant**=**sig**;**

temp**->**sig**=**aux**;**

aux**->**ant**=**temp**;**

aux**=**sig**;**

temp**=**temp**->**sig**;**

**while(**temp**!=**primero**){**

temp**->**setColumnas**(**temp**->**ant**->**getColumnas**()+**1**);**

temp**=**temp**->**sig**;**

**}**

**if(**ultimo**->**getColumnas**()>**10**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}else** **if(**column**>**aux**->**getColumnas**()){**

**while(**aux**->**getColumnas**()+**1**!=**column**){**

aux**=**aux**->**sig**;**

**}**

**if(**aux**->**sig**==**primero**){**

pnodo temp**=new** NodoTetris**(**value**,**numeroFila**(),**ultimo**->**getColumnas**()+**1**);**

ultimo**->**sig**=**temp**;**

temp**->**ant**=**ultimo**;**

temp**->**sig**=**primero**;**

ultimo**=**temp**;**

aux**=**control**;**

**if(**ultimo**->**getColumnas**()>**10**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**}**

aux**=**control**;**

**if(**ultimo**->**getColumnas**()>**10**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**}else{**

pnodo temp**=new** NodoTetris**(**value**,**numeroFila**(),**aux**->**getColumnas**()+**1**);**

sig**=**aux**->**sig**;**

sig**->**ant**=**temp**;**

temp**->**sig**=**sig**;**

temp**->**ant**=**aux**;**

aux**->**sig**=**temp**;**

**while(**temp**!=**primero**){**

temp**->**setColumnas**(**temp**->**ant**->**getColumnas**()+**1**);**

temp**=**temp**->**sig**;**

**}**

aux**=**control**;**

**if(**ultimo**->**getColumnas**()>**10**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**}else{**

**while(**aux**->**getColumnas**()+**1**!=**column**){**

aux**=**aux**->**sig**;**

**}**

pnodo temp**=new** NodoTetris**(**value**,**numeroFila**(),**column**);**

sig**=**aux**->**ant**;**

sig**->**sig**=**temp**;**

temp**->**ant**=**sig**;**

temp**->**sig**=**aux**;**

aux**->**ant**=**temp**;**

aux**=**temp**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**ant**->**getColumnas**()+**1**);**

aux**=**aux**->**sig**;**

**}**

aux**=**control**;**

**if(**ultimo**->**getColumnas**()>**10**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**}**

**}else{**

**while(**aux**->**getColumnas**()!=**column**+**1**){**

aux**=**aux**->**sig**;**

**}**

pnodo temp**=new** NodoTetris**(**value**,**numeroFila**(),**aux**->**ant**->**getColumnas**()+**1**);**

sig**=**aux**->**ant**;**

sig**->**sig**=**temp**;**

temp**->**ant**=**sig**;**

temp**->**sig**=**aux**;**

aux**->**ant**=**temp**;**

aux**=**primero**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**ant**->**getColumnas**()+**1**);**

aux**=**aux**->**sig**;**

**}**

**if(**ultimo**->**getColumnas**()>**10**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**getColumnas**()-**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**}**

**}**

**}**

void ListaCD**::**mostrar**(){**

**if(**listaVacia**()){**

**}else{**

pnodo aux**;**

aux **=** primero**;**

int idC**=**aux**->**getColumnas**();**

int idF**=**aux**->**getFilas**();**

**do{**

aux**->**imprimir**();**

aux**=**aux**->**sig**;**

**}while(**aux**->**getColumnas**()!=**idC**||**aux**->**getFilas**()!=**idF**);**

cout**<<**endl**<<**endl**;**

**}**

**}**

int ListaCD**::**numeroElementos**(){**

pnodo aux**;**

aux **=** primero**;**

int idC**=**aux**->**getColumnas**();**

int idF**=**aux**->**getFilas**();**

int i**=**0**;**

**if(**primero**==NULL){**

**return** 0**;**

**}**

**if(**primero**==**ultimo**){**

**return** 1**;**

**}**

**do{**

i**++;**

aux**=**aux**->**sig**;**

**}while(**aux**->**getColumnas**()!=**idC**||**aux**->**getFilas**()!=**idF**);**

**return** i**;**

**}**

int ListaCD**::**chequeoEliminar**(**int puntaje**){**

**if(!**listaVacia**()){**

unirLista**();**

int p**=**1**;**

pnodo aux**=**primero**;**

pnodo temp**;**

**if(**numeroElementos**()>=**2**){**

**if(**numeroElementos**()==**2**){**

**if(**aux**->**getValorTetris**()==**aux**->**sig**->**getValorTetris**()){**

**delete** primero**;**

**delete** ultimo**;**

primero**=NULL;**

ultimo**==NULL;**

**}**

**}**

**else{**

**while(**aux**!=**ultimo**){**

temp**=**aux**->**sig**;**

**if(**aux**->**getValorTetris**()==**temp**->**getValorTetris**()&&**aux**->**getFilas**()==**temp**->**getFilas**()){**

**if(**aux**==**primero**){**

primero**=**aux**->**sig**->**sig**;**

ultimo**->**sig**=**primero**;**

pnodo aux1**=**aux**;**

pnodo aux2**=**aux**->**sig**;**

aux**=**aux**->**sig**->**sig**;**

**delete** aux1**;**

**delete** aux2**;**

**}else{**

**if(**aux**->**sig**==**ultimo**){**

ultimo**=**aux**->**ant**;**

**}**

aux**->**ant**->**sig**=**temp**->**sig**;**

temp**->**sig**->**ant**=**aux**->**ant**;**

pnodo aux1**=**aux**;**

pnodo aux2**=**aux**->**sig**;**

aux**=**aux**->**sig**->**sig**;**

**delete** aux1**;**

**delete** aux2**;**

**}**

unirLista**();**

puntaje**+=**100**;**

chequeoEliminar**(**puntaje**);**

**}**

aux**=**aux**->**sig**;**

**}**

aux**=**primero**;**

**while(**aux**!=**ultimo**){**

temp**=**aux**->**sig**;**

**while(**temp**!=**ultimo**){**

**if(**aux**->**getColumnas**()==**temp**->**getColumnas**()&&**aux**->**getFilas**()+**1**==**temp**->**getFilas**()){**

**if(**aux**->**getValorTetris**()==**temp**->**getValorTetris**()){**

**if(**aux**==**primero**){**

**if(**temp**==**ultimo**){**

pnodo aux1**=**primero**;**

pnodo aux2**=**ultimo**;**

primero**=**aux**->**sig**;**

ultimo**=**temp**->**ant**;**

ultimo**->**sig**=**primero**;**

temp**=**ultimo**->**ant**;**

aux**=**primero**;**

**delete** aux1**;**

**delete** aux2**;**

**}else{**

pnodo aux1**=**primero**;**

pnodo aux2**=**temp**;**

primero**=**aux**->**sig**;**

ultimo**->**sig**=**primero**;**

temp**->**ant**->**sig**=**temp**->**sig**;**

temp**->**sig**->**ant**=**temp**->**ant**;**

temp**=**temp**->**ant**;**

aux**=**primero**;**

**delete** aux1**;**

**delete** aux2**;**

**}**

**}else{**

**if(**temp**==**ultimo**){**

pnodo aux1**=**aux**;**

pnodo aux2**=**ultimo**;**

ultimo**=**temp**->**ant**;**

ultimo**->**sig**=**primero**;**

aux**->**ant**->**sig**=**aux**->**sig**;**

aux**->**sig**->**ant**=**aux**->**ant**;**

aux**=**aux**->**ant**;**

temp**=**ultimo**->**ant**;**

**delete** aux1**;**

**delete** aux2**;**

**}else{**

pnodo aux1**=**aux**;**

pnodo aux2**=**temp**;**

temp**->**ant**->**sig**=**temp**->**sig**;**

temp**->**sig**->**ant**=**temp**->**ant**;**

aux**->**sig**->**ant**=**aux**->**ant**;**

aux**->**ant**->**sig**=**aux**->**sig**;**

aux**=**aux**->**ant**;**

temp**=**temp**->**ant**;**

**delete** aux1**;**

**delete** aux2**;**

**}**

**}**

unirLista**();**

puntaje**+=**100**;**

chequeoEliminar**(**puntaje**);**

**}**

**}**

temp**=**temp**->**sig**;**

**}**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**return** puntaje**;**

**}**

**}**

int ListaCD**::**numeroFila**(){**

**return** **(**numeroElementos**()/**10**+**1**);**

**}**

/\*void ListaCD::ordenarLista(){

if(!listaVacia()){

pnodo control=primero;

while(control->getFilas()!=1){

primero=control->sig;

ultimo->sig=control;

control->ant=ultimo;

control->sig=primero;

ultimo=control;

control=primero;

}

int i=1, f=1, row, ordenamiento=numeroElementos();

bool flag=true, flag2;

int c=numeroElementos()%10;

if(c!=0){

row=numeroFila();

}else{

row=numeroElementos()/10;

}

while(row>0){

pnodo aux=control;

if(ordenamiento>=10){

if(flag==true){

if(aux->getColumnas()!=1){

aux->setColumnas(1);

}

flag=false;

}

}

while(i!=10){

if(aux->getFilas()==f){

if(aux->sig->getColumnas()!=aux->getColumnas()+1&&aux->sig->getFilas()==aux->getFilas()){

aux->sig->setColumnas(aux->getColumnas()+1);

}

i++;

aux=aux->sig;

}

else{

aux->ant->sig=aux->sig;

aux->sig->ant=aux->ant;

ultimo->sig=aux;

aux->ant=ultimo;

aux->sig=primero;

ultimo=aux;

aux=control;

i=1;

}

if(i==10){

pnodo temp=aux->sig;

if(temp->getColumnas()>temp->sig->getColumnas()&&temp->getFilas()==temp->sig->getFilas()){

temp->ant->sig=temp->sig;

temp->sig->ant=temp->ant;

ultimo->sig=temp;

temp->ant=ultimo;

temp->sig=primero;

ultimo=temp;

}

row--;

control=aux->sig;

flag=true;

f++;

ordenamiento-=10;

}

if(aux->sig==primero||ordenamiento==1){

row--;

i=10;

}

}

i=1;

}

pnodo aux=primero;

while(aux!=ultimo){

if(aux->sig->getColumnas()>10){

aux->sig->setColumnas(aux->sig->getColumnas()-10);

}

aux=aux->sig;

}

pnodo ayuda;

aux=primero;

while(aux!=ultimo){

if(aux->getColumnas()>aux->sig->getColumnas()&&aux->getFilas()==aux->sig->getFilas()){

aux->ant->sig=aux->sig;

aux->sig->ant=aux->ant;

ayuda=aux->sig;

while(ayuda->sig->getFilas()==aux->getFilas()){

ayuda=ayuda->sig;

}

ayuda->sig->ant=aux;

aux->sig=ayuda->sig;

ayuda->sig=aux;

aux->ant=ayuda;

}

aux=aux->sig;

}

}

}\*/

void ListaCD**::**unirLista**(){**

mostrar**();**

**if(!**listaVacia**()){**

**if(**numeroElementos**()<**10**){**

pnodo aux**=**primero**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**ant**->**getColumnas**()+**1**);**

aux**->**setFilas**(**1**);**

aux**=**aux**->**sig**;**

**}**

**}else{**

pnodo control**=**primero**;**

pnodo aux**=**primero**;**

int filas**=**1**;**

int help**=**numeroElementos**();**

**while(**help**>**10**){**

aux**->**setColumnas**(**1**);**

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

control**=**control**->**sig**;**

**}**

aux**=**aux**->**sig**;**

**while(**aux**!=**control**){**

int p**=**aux**->**ant**->**getColumnas**();**

**if(**p**>=**10**){**

p**=**0**;**

**}**

aux**->**setColumnas**(**p**+**1**);**

aux**->**setFilas**(**filas**);**

aux**=**aux**->**sig**;**

**}**

filas**++;**

aux**=**control**;**

help**-=**10**;**

**if(**help**>=**10**){**

aux**->**setColumnas**(**1**);**

**}**

**}**

aux**=**aux**->**sig**;**

**while(**aux**!=**primero**){**

aux**->**setColumnas**(**aux**->**ant**->**getColumnas**()+**1**);**

aux**=**aux**->**sig**;**

**}**

**}**

**}**

**}**

pnodo ListaCD**::**returnNodo**(**int i**){**

pnodo aux**=**primero**;**

**if(**i**==**1**){**

**return** primero**;**

**}**

**for(**int x**=**1**;**x**<**i**;**x**++){**

aux**=**aux**->**sig**;**

**}**

**return** aux**;**

**}**

void ListaCD**::**generarArchivo**(){**

pnodo aux**=**primero**;**

ofstream file**;**

file**.**open**(**"Lista.txt"**);**

file**<<**"Lista Jugadores"**<<**"\n"**;**

**do{**

file**<<**"Valor: "**<<**aux**->**getValorTetris**()<<**" Filas: "**<<**aux**->**getFilas**()<<**" Columnas: "**<<**aux**->**getColumnas**()<<**"\n"**;**

aux**=**aux**->**sig**;**

**}while(**aux**!=**primero**);**

file**.**close**();**

**}**

//CLASE NODOTETRIS

#include <iostream>

#include <stdio.h>

**using** **namespace** std**;**

class NodoTetris**{**

private**:**

int valorTetris**;**

int filas**;**

int columna**;**

NodoTetris**\*** sig**;**

NodoTetris**\*** ant**;**

public**:**

NodoTetris**(**int**,**int**,**int**);**

int getValorTetris**();**

void setValorTetris**(**int**);**

int getColumnas**();**

void setColumnas**(**int**);**

int getFilas**();**

void setFilas**(**int**);**

NodoTetris**\*** getSiguiente**();**

void setSiguiente**(**NodoTetris**\*);**

NodoTetris**\*** getAnterior**();**

void setAnterior**(**NodoTetris**\*);**

void imprimir**();**

friend class ListaCD**;**

**};**

NodoTetris**::**NodoTetris**(**int value**,** int row**,** int column**){**

valorTetris**=**value**;**

filas**=**row**;**

columna**=**column**;**

sig**=NULL;**

ant**=NULL;**

**}**

int NodoTetris**::**getValorTetris**(){**

**return** valorTetris**;**

**}**

void NodoTetris**::**setValorTetris**(**int value**){**

valorTetris**=**value**;**

**}**

int NodoTetris**::**getFilas**(){**

**return** filas**;**

**}**

void NodoTetris**::**setFilas**(**int row**){**

filas**=**row**;**

**}**

int NodoTetris**::**getColumnas**(){**

**return** columna**;**

**}**

void NodoTetris**::**setColumnas**(**int column**){**

columna**=**column**;**

**}**

NodoTetris**\*** NodoTetris**::**getAnterior**(){**

**return** ant**;**

**}**

void NodoTetris**::**setAnterior**(**NodoTetris**\*** before**){**

ant**=**before**;**

**}**

NodoTetris**\*** NodoTetris**::**getSiguiente**(){**

**return** sig**;**

**}**

void NodoTetris**::**setSiguiente**(**NodoTetris**\*** after**){**

sig**=**after**;**

**}**

void NodoTetris**::**imprimir**(){**

cout**<<**valorTetris**;**

cout**<<**" "**<<**filas**;**

cout**<<**" "**<<**columna**<<**endl**;**

**}**

//CLASE TETRISGRAFICA

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <winbgim.h>

#include <windows.h>

#include <string>

#include <mmsystem.h>

#include <sstream>

#include "ListaCD.h"

class TetrisGrafica**{**

public**:**

int grafica**();**

**};**

int TetrisGrafica**::**grafica**(){**

char soundfile**[]** **=** "C:/Users/JORGE GALARZA/Desktop/Proyecto/Proyecto Segundo Parcial/Codigo/Tetris99.wav"**;**

PlaySound**((**LPCSTR**)**soundfile**,** **NULL,** SND\_FILENAME **|** SND\_ASYNC **);**

ListaCD list**;**

srand**(**time**(NULL));**

int y**=**1**,**j**=**550**,** filas**=**0**,** puntaje**=**0**;**

initwindow**(**1300**,**650**);**

rectangle**(**25**,**25**,**1275**,**625**);**

settextstyle**(**8**,** 0**,** 4**);**

setfillstyle**(**1**,** 0**);**

char c**;**

char**\*** cN**=new** char**[**1**];**

int x**=**rand**()%**10**;**

string num**;**

stringstream out**;**

int posx**=**600**,**posy**=**50**;**

sprintf**(**cN**,**"%d"**,**x**);**

**while(**c**!=**27**||**list**.**numeroFila**()>=**10**){**

**while(!**kbhit**()){**

setcolor**(**x**+**1**);**

outtextxy**(**posx**,**posy**,**cN**);**

Sleep**(**250**);**

setcolor**(**15**);**

rectangle**(**posx**-**5**,**posy**-**5**,**posx**+**20**,**posy**+**40**);**

setcolor**(**0**);**

floodfill**(**posx**,**posy**,**15**);**

rectangle**(**posx**-**5**,**posy**-**5**,**posx**+**20**,**posy**+**40**);**

setcolor**(**3**);**

posy**+=**50**;**

**if(**posy**>(**550**-(**filas**\***50**))){**

int column**=**posx**/**120**;**

list**.**insertarEntre**(**x**,**column**);**

// list.mostrar();

**if(**list**.**numeroElementos**()>=**2**){**

puntaje**=**list**.**chequeoEliminar**(**puntaje**);**

// list.mostrar();

**}**

filas**=**list**.**numeroFila**();**

posy**=**50**;**

posx**=**600**;**

free**(**cN**);**

cN**=new** char**[**1**];**

x**=**rand**()%**10**;**

sprintf**(**cN**,**"%d"**,**x**);**

setcolor**(**0**);**

floodfill**(**600**,**325**,**15**);**

**for(**int i**=**1**;**i**<=**list**.**numeroElementos**();**i**++){**

pnodo aux**=**list**.**returnNodo**(**i**);**

char**\*** temp **=** **new** char**[**1**];**

sprintf**(**temp**,**"%d"**,**aux**->**getValorTetris**());**

setcolor**(**aux**->**getValorTetris**()+**1**);**

outtextxy**(**aux**->**getColumnas**()\***120**,**550**-((**aux**->**getFilas**()-**1**)\***50**),**temp**);**

free**(**temp**);**

**}**

**}**

**}**

c**=**getch**();**

**switch(**c**){**

**case** 75**:**

**if(**posx**==**120**)**

posx**=**120**;**

**else**

posx**-=**120**;**

**break;**

**case** 77**:**

**if(**posx**==**1200**)**

posx**=**1200**;**

**else**

posx**+=**120**;**

**break;**

**}**

**}**

PlaySound**(NULL,** **NULL,** 0**);**

list**.**generarArchivo**();**

**return** puntaje**;**

**}**

//CLASE MENU

/\* \*\*\*\*\* UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE \*\*\*\*\* \*\*

\*\* Estructura de Datos \*\*

\*\* Nombre: Jorge Galarza - Kevin Zurita \*\*

\*\* NRC: 2742 \*\*

\*\* Fecha de realizacion: 01/12/2019 \*\*

\*\* Fecha de modificacion: 12/12/2019 \*\*

\*\* Ing. Fernando Solis \*\*

\*/

#include <iostream>

#include <winbgim.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

#include <fstream>

#define TECLA\_DERECHA 77

#define TECLA\_IZQUIERDA 75

#define ENTER 13

#define F1 59

void**\*** selloEspe**(**void **\*);**

static HWND hConWnd**;**

HWND BCX\_Bitmap**(**char**\*,** HWND **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**,** int **=** 0**);**

HWND GetConsoleWndHandle**(**void**);**

**using** **namespace** std**;**

// Funcion gotoxy

void gotoxy**(**int x**,**int y**){**

HANDLE hcon**;**

hcon **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

COORD dwPos**;**

dwPos**.**X **=** x**;**

dwPos**.**Y**=** y**;**

SetConsoleCursorPosition**(**hcon**,**dwPos**);**

**}**

// Marquesina

void**\*** marquesina**(**void **\***data**){**

int a**,** b**,** c**,** n**=**0**,**letra**=**39**,**pos**=**1**,**cont**=**39**,**aux**,**cont1**=**39**,**auxg**;**

char t**[**50**]** **=**"TETRIS: Proyecto Estructura de Datos "**,**auxt**[**39**]=**" "**;**

**do{**

**for** **(**a**=**0**;**a**<**42**;**a**++){**

aux**=**pos**;**

**for(**b**=**39**;**b**>**cont**;**b**--){**

gotoxy**(**pos**,**1**);**

cout**<<**t**[**b**];**

pos**--;**

**}**

aux**++;**

pos**=**aux**;**

cont**--;**

Sleep **(**75**);**

**if(**a**==**40**){**

**break;**

**}**

**}**

**for(**a**=**3**;**a**<**70**;**a**++){**

gotoxy**(**a**-**1**,**1**);**

cout**<<**" "**;**

gotoxy**(**a**,**1**);**

cout**<<**t**;**

Sleep **(**75**);**

**}**

pos**=**70**;**

auxg**=**69**;**

**for** **(**a**=**0**;**a**<**41**;**a**++){**

gotoxy**(**auxg**,**1**);**

cout**<<**" "**;**

aux**=**pos**;**

**for(**b**=**0**;**b**<=**cont1**;**b**++){**

gotoxy**(**pos**,**1**);**

cout**<<**t**[**b**];**

pos**++;**

**}**

cont1**--;**

aux**++;**

pos**=**aux**;**

auxg**++;**

Sleep **(**75**);**

**}**

cont1**=**39**;**

letra**=**39**;**

pos**=**1**;**

cont**=**39**;**

c**=**0**;**

**}while** **(**c**=**1**);**

**}**

// Cambia el color de la letras y dondo de la consola

void SetConsoleColour**(**WORD**\*** Attributes**,** DWORD Colour**){**

CONSOLE\_SCREEN\_BUFFER\_INFO Info**;**

HANDLE hStdout **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);**

GetConsoleScreenBufferInfo**(**hStdout**,** **&**Info**);**

**\***Attributes **=** Info**.**wAttributes**;**

SetConsoleTextAttribute**(**hStdout**,** Colour**);**

**}**

//

void ResetConsoleColour**(**WORD Attributes**){**

SetConsoleTextAttribute**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** Attributes**);**

**}**

// Remarca el metodo uno en la consola

void uno **(**WORD Attributes**){**

cout **<<** endl**;**

SetConsoleColour**(&**Attributes**,** FOREGROUND\_BLUE **|** BACKGROUND\_INTENSITY **|** BACKGROUND\_GREEN**);**

cout**<<**"\tJugar "**;**

ResetConsoleColour**(**Attributes**);**

cout**<<**"\t CodigoQR \t Codigo de Barras \t Sello Pixeleado \t Salir "**<<**endl**;**

**}**

// Remarca el metodo dos en la consola

void dos**(**WORD Attributes**){**

cout **<<** endl**;**

cout**<<**"\tJugar \t "**;**

SetConsoleColour**(&**Attributes**,** FOREGROUND\_BLUE **|** BACKGROUND\_INTENSITY **|** BACKGROUND\_GREEN**);**

cout**<<**"CodigoQR "**;**

ResetConsoleColour**(**Attributes**);**

cout**<<**"\t Codigo de Barras \t Sello Pixeleado \t Salir"**<<**endl**;**

**}**

// Remarca el metodo tres en la consola

void tres**(**WORD Attributes**){**

cout **<<** endl**;**

cout**<<**"\tJugar \t CodigoQR \t "**;**

SetConsoleColour**(&**Attributes**,** FOREGROUND\_BLUE **|** BACKGROUND\_INTENSITY **|** BACKGROUND\_GREEN**);**

cout**<<**"Codigo de Barras"**;**

ResetConsoleColour**(**Attributes**);**

cout**<<**"\t Sello Pixeleado \t Salir"**;**

**}**

// Remarca el metodo cuatro en la consola

void cuatro**(**WORD Attributes**){**

cout **<<** endl**;**

cout**<<**"\tJugar \t CodigoQR \t Codigo de Barras \t "**;**

SetConsoleColour**(&**Attributes**,** FOREGROUND\_BLUE **|** BACKGROUND\_INTENSITY **|** BACKGROUND\_GREEN**);**

cout**<<**"Sello Pixeleado"**;**

ResetConsoleColour**(**Attributes**);**

cout **<<** "\t Salir"**;**

**}**

// Remarca el metodo cinco en la consola

void cinco**(**WORD Attributes**){**

cout **<<** endl**;**

cout**<<**"\tJugar \t CodigoQR \t Codigo de Barras \t Sello Pixeleado \t"**;**

SetConsoleColour**(&**Attributes**,** FOREGROUND\_BLUE **|** BACKGROUND\_INTENSITY **|** BACKGROUND\_GREEN**);**

cout**<<**"Salir"**;**

ResetConsoleColour**(**Attributes**);**

**}**

// funcion principal

int main**()**

**{**

system**(**"color e"**);**

// declaracoion de variables

WORD Attributes**=**0**;**

pthread\_t thread1**;**

pthread\_create**(&**thread1**,** **NULL,** marquesina**,** **NULL);**

int met**,**tecla**,**opc**=**0**;**

cout**<<**"\n\n"**<<**endl**;**

cout**<<**"\t Seleccione el metodo que desea usar \t\t\t\t\t\t Pulse F1 para AYUDA"**<<**endl**;**

cout **<<** endl**;**

cout**<<**"\tJugar \t CodigoQR \t Codigo de Barras \t Sello Pixeleado \t Salir "**<<**endl**;**

**do{**

// lee la tecla que se pulsa

**do{**

tecla**=**getch**();**

**}while(**tecla**!=**TECLA\_DERECHA **&&** tecla**!=**TECLA\_IZQUIERDA **&&** tecla**!=**ENTER **&&** tecla**!=**F1**);**

system**(**"cls"**);**

cout**<<**"\n\n"**<<**endl**;**

cout**<<**"\t Seleccione el metodo que desea usar \t\t\t\t pulse F1 para ayuda"**<<**endl**;;**

// Dependiendo de la tecla que se pulse se usa un contador o abre la ventana de ayuda

**switch** **(**tecla**){**

cout **<<**"\n\n"**<<**opc**<<**endl**;**

**case** TECLA\_DERECHA**:**

opc**++;**

**if(**opc**==**6**){**

opc**=**1**;**

**}**

**break;**

**case** TECLA\_IZQUIERDA**:**

opc**--;**

**if(**opc**<=**0**){**

opc**=**5**;**

**}**

**break;**

**case** ENTER**:**

met**=**opc**;**

**break;**

// en caso de presionar la tecla f1 se abre una ventana de ayuda

**case** F1**:**

system**(**"AyudaTetris.chm"**);**

**break;**

**}**

// Dependiendo del contador entra a un caso en el que pinta el metodo que se seleccionara

**switch(**opc**){**

**case** 1**:**

uno**(**Attributes**);**

**break;**

**case** 2**:**

dos**(**Attributes**);**

**break;**

**case** 3**:**

tres**(**Attributes**);**

**break;**

**case** 4**:**

cuatro**(**Attributes**);**

**break;**

**case** 5**:**

cinco**(**Attributes**);**

**break;**

**}**

// Cuando se pulsa enter entra al metodo que se encuentre seleccionado

**switch(**met**){**

// ejecuta jugar

**case** 1**:**

SuspendThread**(&**thread1**);**

system**(**"Tetrisdelamuerte.exe"**);**

system**(**"cls"**);**

opc**=**0**;**

met**=**0**;**

**break;**

// Ejecuta código QR

**case** 2**:**

system**(**"CodigoQR.png"**);**

opc**=**0**;**

met**=**0**;**

**break;**

// Ejecuta codigo de barras

**case** 3**:**

system**(**"CodigodeBarras.png"**);**

opc**=**0**;**

met**=**0**;**

**break;**

// Ejecuta el sello pixeleado

**case** 4**:**

system**(**"cls"**);**

void **\***status**;**

pthread\_t thread2**;**

pthread\_create **(&**thread2 **,** **NULL** **,** selloEspe **,** **(**void **\*)** **&** thread1**);**

pthread\_join**(**thread2**,** **&**status**);**

opc**=**0**;**

met**=**0**;**

**break;**

**}**

**}while(**met**!=**5**);**

// Termina el programa

system**(**"cls"**);**

cout**<<**"\n\n"**<<**endl**;**

cout**<<**"Gracias por Jugar !!!"**<<**endl**;**

system**(**"PAUSE"**);**

**return** 0**;**

**}**

HWND GetConsoleWndHandle**(**void**)**

**{**

HWND hConWnd**;**

OSVERSIONINFO os**;**

char szTempTitle**[**64**],** szClassName**[**128**],** szOriginalTitle**[**1024**];**

os**.**dwOSVersionInfoSize **=** **sizeof(**OSVERSIONINFO**);**

GetVersionEx**(&**os**);**

// may not work on WIN9x

**if** **(**os**.**dwPlatformId **==** VER\_PLATFORM\_WIN32s**)** **return** 0**;**

GetConsoleTitle**(**szOriginalTitle**,** **sizeof(**szOriginalTitle**));**

sprintf**(**szTempTitle**,** "%u - %u"**,** GetTickCount**(),** GetCurrentProcessId**());**

SetConsoleTitle**(**szTempTitle**);**

Sleep**(**60**);**

// handle for NT and XP

hConWnd **=** FindWindow**(NULL,** szTempTitle**);**

SetConsoleTitle**(**szOriginalTitle**);**

// may not work on WIN9x

**if** **(**os**.**dwPlatformId **==** VER\_PLATFORM\_WIN32\_WINDOWS**)**

**{**

hConWnd **=** GetWindow**(**hConWnd**,** GW\_CHILD**);**

**if** **(**hConWnd **==** **NULL)** **return** 0**;**

GetClassName**(**hConWnd**,** szClassName**,** **sizeof(**szClassName**));**

// while ( \_stricmp( szClassName, "ttyGrab" ) != 0 )

**while** **(**strcmp**(**szClassName**,** "ttyGrab"**)** **!=** 0**)**

**{**

hConWnd **=** GetNextWindow**(**hConWnd**,** GW\_HWNDNEXT**);**

**if** **(**hConWnd **==** **NULL)** **return** 0**;**

GetClassName**(**hConWnd**,** szClassName**,** **sizeof(**szClassName**));**

**}**

**}**

**return** hConWnd**;**

**}**

HWND BCX\_Bitmap**(**char**\*** Text**,** HWND hWnd**,** int id**,** int X**,** int Y**,** int W**,** int H**,** int Res**,** int Style**,** int Exstyle**)**

**{**

HWND A**;**

HBITMAP hBitmap**;**

// set default style

**if** **(!**Style**)** Style **=** WS\_CLIPSIBLINGS **|** WS\_CHILD **|** WS\_VISIBLE **|** SS\_BITMAP **|** WS\_TABSTOP**;**

// form for the image

A **=** CreateWindowEx**(**Exstyle**,** "static"**,** **NULL,** Style**,** X**,** Y**,** 0**,** 0**,** hWnd**,** **(**HMENU**)**id**,** GetModuleHandle**(**0**),** **NULL);**

// Text contains filename

hBitmap **=** **(**HBITMAP**)**LoadImage**(**0**,** Text**,** IMAGE\_BITMAP**,** 0**,** 0**,** LR\_LOADFROMFILE **|** LR\_CREATEDIBSECTION**);**

// auto-adjust width and height

**if** **(**W **||** H**)** hBitmap **=** **(**HBITMAP**)**CopyImage**(**hBitmap**,** IMAGE\_BITMAP**,** W**,** H**,** LR\_COPYRETURNORG**);**

SendMessage**(**A**,** **(**UINT**)**STM\_SETIMAGE**,** **(**WPARAM**)**IMAGE\_BITMAP**,** **(**LPARAM**)**hBitmap**);**

**if** **(**W **||** H**)** SetWindowPos**(**A**,** HWND\_TOP**,** X**,** Y**,** W**,** H**,** SWP\_DRAWFRAME**);**

**return** A**;**

**}**

void**\*** selloEspe**(**void **\***arg**)**

**{**

pthread\_mutex\_t ptmutex1**;**

pthread\_mutex\_lock**(&**ptmutex1**);**

hConWnd **=** GetConsoleWndHandle**();**

BCX\_Bitmap**(**"logo.bmp"**,** hConWnd**,** 123**,** 150**,** 150**,** 0**,** 0**);**

Sleep**(**5000**);**

pthread\_mutex\_unlock**(&**ptmutex1**);**

**}**