```
#include <cstdlib>
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <iomanip>
#include <windows.h>
#include <String.h>
#include "valoresNormales.h"

#define PROBABILIDAD_NEUTRO 0.507
#define PROBABILIDAD_NINIO 0.310
#define PROBABILIDAD_NINIA 0.183
#define PROBABILIDAD_NINIO_DEBIL 0.364
#define PROBABILIDAD_NINIO_MODERADO 0.272
#define PROBABILIDAD_NINIO_SEVERO 0.364
#define PROBABILIDAD_NINIA_DEBIL 0.230
#define PROBABILIDAD_NINIA_MODERADO 0.385
#define PROBABILIDAD_NINIA_SEVERO 0.385
#define PROBABILIDAD_NINIO_DESPUES_DE_NEUTRO 0.629
#define PROBABILIDAD_NINIA_DESPUES_DE_NEUTRO 0.371

using namespace std;

typedef struct EVENTO{
   int tipo;
   int intensidad;
   int duracion;
   float pico;
   int mesComienzo;
   float valorComienzo;
   float valorFinal;
}Evento;

typedef struct ESTADISTICA{
   int totalNinios;
   int totalNinias;
   int totalNeutros;
   int totalEventos;
   int mesesSimulados;
   float ninioMasElevado;
   float ninioMasBajo;
   float niniaMasElevada;
   float niniaMasBaja;
   int neutralMasLargo;
   int neutralMasCorto;
   int ninioMasLargo;
   int ninioMasCorto;
   int niniaMasLarga;
```

```
   int niniaMasCorta;
}Estadistica;

typedef struct NODO_EVENTO{
   Evento evento;
   struct NODO_EVENTO* sig;
} NodoEvento;

typedef struct NODO_ONI{
   float oni;
   struct NODO_ONI* sig;
} NodoOni;

NodoEvento* crearNodoEvento(Evento evento){

   NodoEvento* nuevo = (NodoEvento*) malloc(sizeof(NodoEvento));

   nuevo->evento = evento;
   nuevo->sig = NULL;

   return nuevo;
}

NodoOni* crearNodoOni(float oni){

   NodoOni* nuevo = (NodoOni*) malloc(sizeof(NodoOni));

   nuevo->oni = oni;
   nuevo->sig = NULL;

   return nuevo;
}

NodoEvento* insertarNodoEvento(NodoEvento* lista, Evento evento){

   NodoEvento* nuevo = crearNodoEvento(evento);

   NodoEvento* aux = lista;

   while( aux && aux->sig ){
      aux = aux->sig;
   }

   if (lista){
      aux->sig = nuevo;
      return lista;
   }else
      return nuevo;
}
```

```
NodoOni* insertarNodoOni(NodoOni* lista, float oni){

   NodoOni* nuevo = crearNodoOni(oni);

   NodoOni* aux = lista;

   while( aux && aux->sig ){
      aux = aux->sig;
   }

   if (lista){
      aux->sig = nuevo;
      return lista;
   }else
      return nuevo;
}

void liberarListaEventos(NodoEvento** lista){

   if (*lista){
      liberarListaEventos( &((*lista)->sig) );

      free(*lista);

      *lista = NULL;
   }
}

void liberarListaOnis(NodoOni** lista){

   if (*lista){
      liberarListaOnis( &((*lista)->sig) );

      free(*lista);

      *lista = NULL;
   }
}

int    ninioDebilDuracion[700];
float  ninioDebilPico[700];
int    ninioDebilMesComienzo[700];
float  ninioDebilValorComienzo[700];
float  ninioDebilValorFinal[700];

int    ninioModeradoDuracion[700];
float  ninioModeradoPico[700];
int    ninioModeradoMesComienzo[700];
```

```
float   ninioModeradoValorComienzo[700];
float   ninioModeradoValorFinal[700];

int     ninioSeveroDuracion[700];
float   ninioSeveroPico[700];
int     ninioSeveroMesComienzo[700];
float   ninioSeveroValorComienzo[700];
float   ninioSeveroValorFinal[700];

int     niniaDebilDuracion[700];
float   niniaDebilPico[700];
int     niniaDebilMesComienzo[700];
float   niniaDebilValorComienzo[700];
float   niniaDebilValorFinal[700];

int     niniaModeradoDuracion[700];
float   niniaModeradoPico[700];
int     niniaModeradoMesComienzo[700];
float   niniaModeradoValorComienzo[700];
float   niniaModeradoValorFinal[700];

int     niniaSeveroDuracion[700];
float   niniaSeveroPico[700];
int     niniaSeveroMesComienzo[700];
float   niniaSeveroValorComienzo[700];
float   niniaSeveroValorFinal[700];

int     neutralDuracion[700];
int     neutralMesComienzo[700];
float   neutralValorComienzoVieneDeNinia[700];
float   neutralValorComienzoVieneDeNinio[700];
float   neutralValorFinalVieneDeNinia[700];
float   neutralValorFinalVieneDeNinio[700];

void inicializarValoresNormales(){
   obtenerValoresNormales(ninioDebilDuracion, ninioDebilPico, ninioDebilMesComienzo, ninioDebilValorComienzo,
ninioDebilValorFinal, ninioModeradoDuracion, ninioModeradoPico, ninioModeradoMesComienzo,
ninioModeradoValorComienzo, ninioModeradoValorFinal, ninioSeveroDuracion,   ninioSeveroPico,
ninioSeveroMesComienzo,   ninioSeveroValorComienzo,   ninioSeveroValorFinal, niniaDebilDuracion,    niniaDebilPico,
niniaDebilMesComienzo,    niniaDebilValorComienzo,    niniaDebilValorFinal, niniaModeradoDuracion,
niniaModeradoPico, niniaModeradoMesComienzo, niniaModeradoValorComienzo, niniaModeradoValorFinal,
niniaSeveroDuracion,   niniaSeveroPico,   niniaSeveroMesComienzo,   niniaSeveroValorComienzo, niniaSeveroValorFinal,
neutralDuracion, neutralMesComienzo, neutralValorComienzoVieneDeNinia, neutralValorComienzoVieneDeNinio,
neutralValorFinalVieneDeNinia, neutralValorFinalVieneDeNinio);
}

void simularEventos(NodoEvento** eventos, int anios){
   int     tipo, intensidad, duracion, mesComienzo;
   float   valorComienzo, valorFinal, pico;
```

```
int     tipoAnterior;
float aleatorioFloat;
int    aleatorioInt;
Evento evento;
int mesesSolicitados = anios*24;
int mesesSimulados = 0;
int i=0;

do{
   aleatorioFloat = (rand() / (double)RAND_MAX);
   if(i==0){
      tipo = 0;
      i++;
   }else{
      if(tipoAnterior==0){
         aleatorioFloat = (rand() / (double)RAND_MAX);
         if(aleatorioFloat <= PROBABILIDAD_NINIO_DESPUES_DE_NEUTRO)
            tipo = 1;
         else
            tipo = 2;
      }else{
         tipo = 0;
      }
   }

   aleatorioFloat = (rand() / (double)RAND_MAX);
   if(tipo==0){//ES neutral
      tipoAnterior=(rand()%2)+1;
      aleatorioInt = rand()%700;
      duracion = neutralDuracion[aleatorioInt];
      aleatorioInt = rand()%700;
      mesComienzo = neutralMesComienzo[aleatorioInt];
      if(tipoAnterior==1){
         aleatorioInt = rand()%700;
         valorComienzo = neutralValorComienzoVieneDeNinio[aleatorioInt];
         aleatorioInt = rand()%700;
         valorFinal = neutralValorFinalVieneDeNinio[aleatorioInt];
      }else{
         if(tipoAnterior==2){
            aleatorioInt = rand()%700;
            valorComienzo = neutralValorComienzoVieneDeNinia[aleatorioInt];
            aleatorioInt = rand()%700;
            valorFinal = neutralValorFinalVieneDeNinia[aleatorioInt];
         }
      }
   }else{
      if(tipo==1){//ES nIÑO
         if(aleatorioFloat<=PROBABILIDAD_NINIO_DEBIL)
            intensidad = 0; //Debil
```

```
    else
      if(aleatorioFloat<=PROBABILIDAD_NINIO_DEBIL+PROBABILIDAD_NINIO_MODERADO)
        intensidad= 1; //moderado
      else
        intensidad = 2; //severo

    if(intensidad==0){ //debil
      aleatorioInt = rand()%700;
      duracion = ninioDebilDuracion[aleatorioInt];
      aleatorioInt = rand()%700;
      pico = ninioDebilPico[aleatorioInt];
      aleatorioInt = rand()%700;
      mesComienzo = ninioDebilMesComienzo[aleatorioInt];
      aleatorioInt = rand()%700;
      valorComienzo = ninioDebilValorComienzo[aleatorioInt];
      aleatorioInt = rand()%700;
      valorFinal = ninioDebilValorFinal[aleatorioInt];
    }else{
      if(intensidad==1){ //Moderado
        aleatorioInt = rand()%700;
        duracion = ninioModeradoDuracion[aleatorioInt];
        aleatorioInt = rand()%700;
        pico = ninioModeradoPico[aleatorioInt];
        aleatorioInt = rand()%700;
        mesComienzo = ninioModeradoMesComienzo[aleatorioInt];
        aleatorioInt = rand()%700;
        valorComienzo = ninioModeradoValorComienzo[aleatorioInt];
        aleatorioInt = rand()%700;
        valorFinal = ninioModeradoValorFinal[aleatorioInt];
      }else{//Severo
        aleatorioInt = rand()%700;
        duracion = ninioSeveroDuracion[aleatorioInt];
        aleatorioInt = rand()%700;
        pico = ninioSeveroPico[aleatorioInt];
        aleatorioInt = rand()%700;
        mesComienzo = ninioSeveroMesComienzo[aleatorioInt];
        aleatorioInt = rand()%700;
        valorComienzo = ninioSeveroValorComienzo[aleatorioInt];
        aleatorioInt = rand()%700;
        valorFinal = ninioSeveroValorFinal[aleatorioInt];
      }
    }

  }else{ //Es Niña
    if(aleatorioFloat<=PROBABILIDAD_NINIA_DEBIL)
      intensidad = 0;
    else
      if(aleatorioFloat<=PROBABILIDAD_NINIA_DEBIL+PROBABILIDAD_NINIA_MODERADO)
        intensidad= 1;
```

```
        else
            intensidad = 2;

    if(intensidad==0){ //debil
        aleatorioInt = rand()%700;
        duracion = niniaDebilDuracion[aleatorioInt];
        aleatorioInt = rand()%700;
        pico = niniaDebilPico[aleatorioInt];
        aleatorioInt = rand()%700;
        mesComienzo = niniaDebilMesComienzo[aleatorioInt];
        aleatorioInt = rand()%700;
        valorComienzo = niniaDebilValorComienzo[aleatorioInt];
        aleatorioInt = rand()%700;
        valorFinal = niniaDebilValorFinal[aleatorioInt];
    }else{
        if(intensidad==1){ //Moderado
            aleatorioInt = rand()%700;
            duracion = niniaModeradoDuracion[aleatorioInt];
            aleatorioInt = rand()%700;
            pico = niniaModeradoPico[aleatorioInt];
            aleatorioInt = rand()%700;
            mesComienzo = niniaModeradoMesComienzo[aleatorioInt];
            aleatorioInt = rand()%700;
            valorComienzo = niniaModeradoValorComienzo[aleatorioInt];
            aleatorioInt = rand()%700;
            valorFinal = niniaModeradoValorFinal[aleatorioInt];
        }else{//Severo
            aleatorioInt = rand()%700;
            duracion = niniaSeveroDuracion[aleatorioInt];
            aleatorioInt = rand()%700;
            pico = niniaSeveroPico[aleatorioInt];
            aleatorioInt = rand()%700;
            mesComienzo = niniaSeveroMesComienzo[aleatorioInt];
            aleatorioInt = rand()%700;
            valorComienzo = niniaSeveroValorComienzo[aleatorioInt];
            aleatorioInt = rand()%700;
            valorFinal = niniaSeveroValorFinal[aleatorioInt];
        }
    }
  }
}

evento.tipo = tipo;
evento.intensidad = intensidad;
evento.duracion = duracion;
evento.mesComienzo = mesComienzo;
evento.valorComienzo = valorComienzo;
evento.valorFinal = valorFinal;
```

```
      if(tipo!=0){
         evento.pico = pico;
      }

      *eventos = insertarNodoEvento(*eventos, evento);
      tipoAnterior=tipo;


      mesesSimulados += duracion;
   }while(mesesSimulados <= mesesSolicitados);
}

void imprimirEventos(NodoEvento* lista){
   Evento evento;

   while(lista){
      evento = lista->evento;
      cout << endl;
      cout << "Tipo: " << evento.tipo << endl;
      cout << "Duracion: " << evento.duracion << endl;
      cout << "MesComienzo: " << evento.mesComienzo << endl;
      cout << "ValorComienzo: " << evento.valorComienzo << endl;
      cout << "ValorFinal: " << evento.valorFinal << endl;
      if(evento.tipo!=0){
         cout << "Intensidad: " << evento.intensidad << endl;
         cout << "Pico: " << evento.pico << endl;
      }
      cout << endl;
      lista = lista->sig;
   }
}

void imprimirOnis(NodoOni* lista, int anios){
   int i=1;
   int n =0;
   cout << setprecision(1) << fixed;

   cout <<" DJF  JFM  FMA  MAM  AMJ  MJJ  JJA  JAS  ASO  SON  OND  NJD";
   cout<<endl;
   while(i<=anios*12){
       if (lista->oni<=-0.5){
         //SetColor(3);
         cout<<lista->oni<<" ";
       }
       else{
          if (lista->oni>=0.5) {
            //SetColor(4);
            cout << " "<<lista->oni << " ";
          }
```

```
            else
            if (lista->oni < 0 && lista->oni >-0.5){
              //SetColor(7);  //Blancos los neutrales negativos
              cout <<lista->oni << " ";
            }
            else{
              //SetColor(7);  //Blancos los neutrales positivos
              cout << " "<<lista->oni << " ";
            }

         }
        lista = lista->sig;

        if(lista){
           if(i%12==0)
              cout<<endl;
        }
        i++;
    }
    //SetColor(7);
}

void inicializarEstadisticas(Estadistica &estadisticas){

    estadisticas.totalNinios = 0;
    estadisticas.totalNinias = 0;
    estadisticas.totalNeutros = 0;
    estadisticas.totalEventos = 0;
    estadisticas.ninioMasElevado = 0;
    estadisticas.ninioMasBajo = 5;
    estadisticas.niniaMasElevada = 0;
    estadisticas.niniaMasBaja = -5;
    estadisticas.neutralMasLargo = 0;
    estadisticas.neutralMasCorto = 50;
    estadisticas.ninioMasLargo = 0;
    estadisticas.ninioMasCorto = 50;
    estadisticas.niniaMasLarga = 0;
    estadisticas.niniaMasCorta = 50;
    estadisticas.mesesSimulados = 0;
}

int contarEventos(NodoEvento* lista){

    int i;

    for( i= 0; lista; i++, lista = lista->sig);

    return i;
}
```

```
int contarOnis(NodoOni* lista){

   int i;

   for( i= 0; lista; i++, lista = lista->sig);

   return i;
}

void imprimirEstadisticas(Estadistica estadisticas){
   cout << endl << endl;
   cout << "Total Ninios: " << estadisticas.totalNinios << endl;
   cout << "Total Ninias: " << estadisticas.totalNinias << endl;
   cout << "Total Neutros: " << estadisticas.totalNeutros << endl;
   cout << "Total Eventos: " << estadisticas.totalEventos << endl;
   cout << "Ninio Mas Elevado: " << estadisticas.ninioMasElevado << endl;
   cout << "Ninio Mas Bajo: " << estadisticas.ninioMasBajo << endl;
   cout << "Ninia Mas Elevada: " << estadisticas.niniaMasElevada << endl;
   cout << "Ninia Mas Baja: " << estadisticas.niniaMasBaja << endl;
   cout << "Neutral Mas Largo: " << estadisticas.neutralMasLargo << endl;
   cout << "Neutral Mas Corto: " << estadisticas.neutralMasCorto << endl;
   cout << "Ninio Mas Largo: " << estadisticas.ninioMasLargo << endl;
   cout << "Ninio Mas Corto: " << estadisticas.ninioMasCorto << endl;
   cout << "Ninia Mas Larga: " << estadisticas.niniaMasLarga << endl;
   cout << "Ninia Mas Corta: " << estadisticas.niniaMasCorta << endl;
}

NodoOni* distribuirEvento(int mesInicio, int duracion, float vInicio, float pico, float vFinal){

   float valor = vInicio;
   float paso;
   float maxPaso;
   int mesPico;
   int mesFinal =mesInicio+duracion-1;
   NodoOni* onis = NULL;

   if(pico==0 && mesInicio==0){ //Distribucion de un netral de niño a niño o niña a niña

      if(duracion<=4)
         maxPaso = (rand() / (double)RAND_MAX)*0.5;
      else
         if(duracion<=8)
            maxPaso = (rand() / (double)RAND_MAX)*0.4;
         else
            maxPaso = (rand() / (double)RAND_MAX)*0.3;

      paso = 1/((float)duracion/4)*maxPaso;
      if(vInicio<0)
```

```
        paso*=-1;
    for (int i = 0; i < duracion; i++) {
        onis = insertarNodoOni(onis, valor);

        if (i < duracion / 2) {
            if ((i != (duracion / 2) - 1) || (duracion % 2 != 0))
                valor -= paso;
        } else {
            valor += paso;
        }
    }
}else
    if(pico==0 && mesInicio==-1){//Distribucion neutral de niño a niña ó niña a niño


        paso = (vInicio-vFinal) / (duracion-1);

        if(vInicio<0 && vFinal>0){
            if(paso<0)
                paso *=-1;
        }else{
            if(paso>0)
                paso*=-1;
        }

        for (int i =0; i < duracion; i++) {
            onis = insertarNodoOni(onis, valor);
            valor += paso;
        }


    }else{
        mesPico = mesInicio + (duracion / 2) - 1;

        if (mesPico < 11)
            mesPico = 11;
        else
            if (mesPico > 13)
            mesPico = 13;

        paso = (pico - vInicio) / (mesPico - mesInicio);

        for (int i = mesInicio; i <= mesPico; i++) {
            onis = insertarNodoOni(onis, valor);
            valor += paso;
        }
        valor -= paso;
        paso = (pico - vFinal) / (mesFinal - mesPico);
```

```
        for (int i = mesPico + 1; i <= mesFinal; i++) {
            valor -= paso;
            onis = insertarNodoOni(onis, valor);
        }
      }
   return onis;
}

float valorInicioNeutral(int tipoAnterior){
   float aleatorio = (rand() / (double)RAND_MAX);
   float vInicio;
   if(aleatorio<(1/7))
      vInicio=0.2;
   else
      if(aleatorio<(3/7))
         vInicio=0.3;
      else
         vInicio=0.4;

   if(tipoAnterior==2)
      vInicio*=-1;

   return vInicio;
}

float valorFinalNeutral(int tipoSiguiente){
   float aleatorio = (rand() / (double)RAND_MAX);
   float vFinal;
   if(aleatorio<(1/7))
      vFinal=0.2;
   else
      if(aleatorio<(3/7))
         vFinal=0.3;
      else
         vFinal=0.4;

   if(tipoSiguiente==2)
      vFinal*=-1;

   return vFinal;
}

NodoOni* unirOnis(NodoOni* onisActuales, NodoOni* onisNuevos){
  NodoOni* aux;
  if (onisActuales == NULL){
    onisActuales = onisNuevos;
  }
  else{
     if (onisNuevos != NULL){
```

```
        aux = onisActuales;
        while (aux->sig != NULL)
          aux = aux->sig;

        aux->sig = onisNuevos;
      }
  }
  return onisActuales;
}

void determinarOnis(NodoEvento* eventos, NodoOni** onis, int anios, Estadistica &estadisticas){
   int mesInicio;
   int duracion;
   float vInicio;
   float pico;
   float vFinal;
   Evento evento;
   int tipoAnterior=-1;
   int mesActual=1;
   NodoEvento* aux = eventos;
   NodoOni* onisNuevos = NULL;
   int duracionAcumulada = 0;
   while(aux){
      evento = aux->evento;

      if(tipoAnterior==-1){
         vInicio=valorInicioNeutral((rand()%2)+1);
         mesInicio=0;
         duracion = aux->sig->evento.mesComienzo-1;
         pico=0;
         vFinal = valorFinalNeutral(aux->sig->evento.tipo);

         if(vInicio*vFinal<0)
            mesInicio=-1;
         else
            mesInicio=0;

         onisNuevos = distribuirEvento(mesInicio, duracion, vInicio, pico, vFinal);
      }else{
         if(tipoAnterior==0){
            mesInicio = evento.mesComienzo;
            duracion = evento.duracion;
            vInicio = evento.valorComienzo;
            pico = evento.pico;
            vFinal = evento.valorFinal;
            onisNuevos = distribuirEvento(mesInicio, duracion, vInicio, pico, vFinal);
         }else{
            pico=0;
            vInicio = valorInicioNeutral(tipoAnterior);
```

```
      if(aux->sig){

         if(aux->sig->evento.mesComienzo-mesActual < 3)
            duracion = 12-mesActual+aux->sig->evento.mesComienzo-1;
         else
            duracion = aux->sig->evento.mesComienzo-mesActual-1;

         vFinal = valorFinalNeutral(aux->sig->evento.tipo);
      }else{
         vFinal = valorFinalNeutral((rand()%2)+1);
         duracion = 12-mesActual;
      }
      if((vInicio>0 && vFinal<0) || ((vInicio<0 && vFinal>0)) )
         mesInicio=-1;
      else
         mesInicio=0;

      onisNuevos = distribuirEvento(mesInicio, duracion, vInicio, pico, vFinal);
   }
}
duracionAcumulada +=duracion;
*onis = unirOnis(*onis, onisNuevos);
mesActual = (duracionAcumulada%12);
if(mesActual==0)
   mesActual=12;

tipoAnterior=evento.tipo;
aux=aux->sig;

//Calculando estadísticas
estadisticas.totalEventos++;
estadisticas.mesesSimulados+= duracion;

if(evento.tipo==0){
   estadisticas.totalNeutros++;

   if(evento.duracion > estadisticas.neutralMasLargo){
      estadisticas.neutralMasLargo = evento.duracion;
   }

   if(evento.duracion < estadisticas.neutralMasCorto){
      estadisticas.neutralMasCorto = evento.duracion;
   }

}else{
   if(evento.tipo==1){
      estadisticas.totalNinios++;

      if(evento.pico > estadisticas.ninioMasElevado)
```

```
            estadisticas.ninioMasElevado = evento.pico;

          if(evento.duracion > estadisticas.ninioMasLargo){
            estadisticas.ninioMasLargo = evento.duracion;
          }

          if(evento.pico < estadisticas.ninioMasBajo)
            estadisticas.ninioMasBajo = evento.pico;

          if(evento.duracion < estadisticas.ninioMasCorto){
            estadisticas.ninioMasCorto = evento.duracion;
          }

        }else{
          estadisticas.totalNinias++;
          if(evento.pico < estadisticas.niniaMasElevada)
            estadisticas.niniaMasElevada = evento.pico;

          if(evento.duracion > estadisticas.niniaMasLarga){
            estadisticas.niniaMasLarga = evento.duracion;
          }

          if(evento.pico > estadisticas.niniaMasBaja)
            estadisticas.niniaMasBaja = evento.pico;

          if(evento.duracion < estadisticas.niniaMasCorta){
            estadisticas.niniaMasCorta = evento.duracion;
          }
        }
      }

    }
}

void escribirOnis(NodoOni* onis, int anios){

    FILE *archivo;
    int neutrales = 0;
    int ninios = 0;
    int ninias = 0;
    int tipoAnterior = -1;
    int tipoActual;
    float maxOniNinio = -5;
    float maxOniNinia = 5;
    float minOniNinio = 5;
    float minOniNinia = -5;
    archivo = fopen ( "onis.txt", "w" );
    if (archivo==NULL){
      cout << "Error al abrir el archivo" << endl;
```

```
   fclose (archivo);
   return ;
}
int n = contarOnis(onis);
fprintf(archivo, "onis=[\n");
int i=0;
while(i<anios*12){

   if(onis->oni<=-0.5){
      tipoActual = 2;

      if(onis->oni<maxOniNinia)
         maxOniNinia = onis->oni;
      if(onis->oni>minOniNinia)
         minOniNinia = onis->oni;
   }else
      if(onis->oni<0.5)
         tipoActual=0;
      else{
         tipoActual=1;

         if(onis->oni>maxOniNinio)
            maxOniNinio = onis->oni;
         if(onis->oni<minOniNinio)
            minOniNinio = onis->oni;
      }

   if(tipoAnterior!=tipoActual){
      if(tipoActual==0){
         neutrales++;
      }
      else{
         if(tipoActual==1){
            ninios++;
         }
         else{
            ninias++;
         }
      }
   }

   tipoAnterior=tipoActual;

   fprintf(archivo, "%f",onis->oni);
   onis = onis->sig;
   if(i+1<anios*12){
      fprintf(archivo, ",\n");

   }
```

```
        i++;
    }

    if(maxOniNinio == -5){
            maxOniNinio = 0;
    }
    if(maxOniNinia == 5){
            maxOniNinia = 0;
    }
    if(minOniNinio == 5){
            minOniNinio = 0;
    }
    if(minOniNinia == -5){
            minOniNinia = 0;
    }

    fprintf(archivo, "];\n");
    fprintf(archivo, "numNinios = %d;\n",ninios);
    fprintf(archivo, "numNinias = %d;\n",ninias);
    fprintf(archivo, "numNeutrales = %d;\n",neutrales);
    fprintf(archivo, "maxOniNinio = %.1f;\n",maxOniNinio);
    fprintf(archivo, "maxOniNinia = %.1f;\n",maxOniNinia*-1);
    fprintf(archivo, "minOniNinio = %.1f;\n",minOniNinio);
    fprintf(archivo, "minOniNinia = %.1f;\n",minOniNinia*-1);
    fclose (archivo);
}

int main(int argc, char** argv) {
    srand(time(NULL));
    int anios;

    Estadistica estadisticas;
    NodoEvento* eventos = NULL;
    NodoOni* onis = NULL;

    cout << "Ingrese anios "<< endl;
    cin >> anios;
    getchar();

    inicializarValoresNormales();
    inicializarEstadisticas(estadisticas);
    simularEventos(&eventos, anios);
    determinarOnis(eventos, &onis, anios, estadisticas);
    imprimirOnis(onis, anios);
    escribirOnis(onis, anios);
    liberarListaOnis(&onis);
    liberarListaEventos(&eventos);
    string link = "index.html";
    ShellExecute(NULL, "open", link.c_str(), NULL, NULL, SW_SHOWNORMAL);
```

```
    getchar();
    return 0;
}
```