```
/*
 * File:   Pedido.h
 * Author: Silvia
 * Silvia Vargas Cáceres
 *
 * Created on 23 de noviembre de 2022, 08:45 PM
 */

#ifndef PEDIDO_H
#define PEDIDO_H

struct Pedido{
    int codigo;
    struct ProductoPedido *productos;
    int cantProductos;
    int hora;
    char *repartidorRappi;
    char *codUsuario;
    char *direccion;
    char *distrito;
    double montoTotal;
};

/*
 * File:   ProductoPedido.h
 * Author: Silvia
 * Silvia Vargas Cáceres
 *
 * Created on 23 de noviembre de 2022, 08:44 PM
 */

#ifndef PRODUCTOPEDIDO_H
#define PRODUCTOPEDIDO_H

struct ProductoPedido{
    char *nombre;
    int cantidad;
    double preUni;
};

#endif /* PRODUCTOPEDIDO_H */

/*
 * File:   Pedido.h
 * Author: Silvia
 * Silvia Vargas Cáceres
 *
 * Created on 23 de noviembre de 2022, 08:45 PM
 */

#ifndef PEDIDO_H
#define PEDIDO_H

struct Pedido{
    int codigo;
    struct ProductoPedido *productos;
    int cantProductos;
    int hora;
    char *repartidorRappi;
    char *codUsuario;
    char *direccion;
    char *distrito;
    double montoTotal;
};

#endif /* PEDIDO_H */

/*
 * File:   funciones.h
```

```
 * Author: Silvia
 * Silvia Vargas Cáceres
 *
 * Created on 23 de noviembre de 2022, 08:47 PM
 */

#ifndef FUNCIONES_H
#define FUNCIONES_H

void cargarDirecciones(struct Direccion *arrDirecciones,int &cantDirecciones);
void cargarPedidos(struct Pedido *arrPedidos,int &cantPedidos);
void completarPedidos(struct Pedido *arrPedidos,int cantPedidos,
        struct Direccion *arrDirecciones,int cantDirecciones);
int buscarUsuario(char *codUsuario,struct Direccion *arrDirecciones,int cantDirecciones);
int buscarPedido(int pedido,struct Pedido *arrPedidos,int cantPedidos);
void asignarProductosaPedido(struct Pedido &pedidoDat,ifstream &archPedidos);
void emitirReporte(struct Pedido *arrPedidos,int cantPedidos);
void imprimirPedido(struct Pedido pedidoDat,ofstream &archReporte);
void imprimirLinea(char car, int cant, ofstream &archivo);
#endif /* FUNCIONES_H */

/*
 * File:   main.cpp
 * Author: Silvia
 * Silvia Vargas Cáceres
 *
 * Created on 23 de noviembre de 2022, 08:38 PM
 */

#include <iostream>
#include <fstream>
#include <iomanip>

using namespace std;

#include "Direccion.h"
#include "ProductoPedido.h"
#include "Pedido.h"
#include "funciones.h"

#define MAX_DIRECCIONES 50
#define MAX_PEDIDOS 100

int main(int argc, char** argv) {
    struct Direccion arrDirecciones[MAX_DIRECCIONES]{};
    struct Pedido arrPedidos[MAX_PEDIDOS]={};
    int cantDirecciones,cantPedidos;
    cargarDirecciones(arrDirecciones,cantDirecciones);
    cargarPedidos(arrPedidos,cantPedidos);
    completarPedidos(arrPedidos,cantPedidos,arrDirecciones,cantDirecciones);
    emitirReporte(arrPedidos,cantPedidos);
    return 0;
}

*
 * File:   funciones.cpp
 * Author: Silvia
 * Silvia Vargas Cáceres
 *
 * Created on 23 de noviembre de 2022, 08:47 PM
 */

#include <iostream>
#include <fstream>
#include <iomanip>
#include <cstring>

using namespace std;

```

```
#include "Direccion.h"
#include "ProductoPedido.h"
#include "Pedido.h"
#include "funciones.h"

#define MAX_PRODUCTOS 10
#define NO_ENCONTRADO -1
#define TAM_LINEA 100

void cargarDirecciones(struct Direccion *arrDirecciones,int &cantDirecciones){
    ifstream archDirecciones("direcciones.csv",ios::in);
    if (not archDirecciones.is_open()){
        cout<<"Error al abrir el archivo direcciones.csv"<<endl;
        exit(1);
    }
    struct Direccion direccionDat;
    cantDirecciones=0;
    char codUsuario[7],direccion[50],distrito[30];
    while(true){
        archDirecciones.getline(codUsuario,7,',');
        if (archDirecciones.eof()) break;
        archDirecciones.getline(direccion,50,',');
        archDirecciones.getline(distrito,50,',');
        direccionDat.codUsuario=new char[strlen(codUsuario)+1];
        strcpy(direccionDat.codUsuario,codUsuario);
        direccionDat.direccion=new char[strlen(direccion)+1];
        strcpy(direccionDat.direccion,direccion);
        direccionDat.distrito=new char[strlen(distrito)+1];
        strcpy(direccionDat.distrito,distrito);
        arrDirecciones[cantDirecciones]=direccionDat;
        cantDirecciones++;
        while(archDirecciones.get()!='\n');
    }
}

void cargarPedidos(struct Pedido *arrPedidos,int &cantPedidos){
    ifstream archPedidos("pedidos.csv",ios::in);
    if (not archPedidos.is_open()){
        cout<<"Error al abrir el archivo pedidos.csv"<<endl;
        exit(1);
    }
    cantPedidos=0;
    struct Pedido pedidoDat;
    int codigo,posPedido;
    while(true){
        archPedidos>>codigo;
        if (archPedidos.eof()) break;
        archPedidos.get();
        posPedido=buscarPedido(codigo,arrPedidos,cantPedidos);
        if (posPedido==NO_ENCONTRADO){
            pedidoDat.codigo=codigo;
            pedidoDat.productos=new struct ProductoPedido[MAX_PRODUCTOS];
            pedidoDat.cantProductos=0;
            pedidoDat.montoTotal=0;
            posPedido=cantPedidos;
            cantPedidos++;
        }
        else
            pedidoDat=arrPedidos[posPedido];
        asignarProductosaPedido(pedidoDat,archPedidos);
        arrPedidos[posPedido]=pedidoDat;
    }
}


void asignarProductosaPedido(struct Pedido &pedidoDat,ifstream &archPedidos){
    double monto=0;
    while(true){
        struct ProductoPedido productoDat;
```

```
208          char nombProd[50];
209          if (pedidoDat.cantProductos<10){
210              archPedidos.getline(nombProd,50,',');
211              archPedidos>>productoDat.cantidad;
212              archPedidos.get();
213              archPedidos>>productoDat.preUni;
214              productoDat.nombre=new char[strlen(nombProd)+1];
215              strcpy(productoDat.nombre,nombProd);
216              pedidoDat.productos[pedidoDat.cantProductos]=productoDat;
217              monto+=(productoDat.preUni*productoDat.cantidad);
218              pedidoDat.cantProductos++;
219              if (archPedidos.get()=='\n') break;
220          }
221          else{
222              while(archPedidos.get()!='\n');
223              break;
224          }
225      }
226      pedidoDat.montoTotal+=monto;
227  }
228
229
230  int buscarUsuario(char *codUsuario,struct Direccion *arrDirecciones,int cantDirecciones){
231      for (int i=0;i<cantDirecciones;i++)
232          if (strcmp(codUsuario,arrDirecciones[i].codUsuario)==0) return i;
233      return NO_ENCONTRADO;
234  }
235
236  void completarPedidos(struct Pedido *arrPedidos,int cantPedidos,
237          struct Direccion *arrDirecciones,int cantDirecciones){
238      ifstream archRappi("rappi.csv",ios::in);
239      if (not archRappi.is_open()){
240          cout<<"Error al abrir el archivo rappi.csv"<<endl;
241          exit(1);
242      }
243      int hh,mm,hora,pedido,posUsuario,posPedido;
244      char car,repartidor[50],codUsuario[7];
245      while(true){
246          archRappi>>hh;
247          if (archRappi.eof()) break;
248          archRappi>>car>>mm;
249          hora=hh*60+mm;
250          archRappi.get();
251          archRappi.getline(repartidor,50,',');
252          archRappi.getline(codUsuario,7,',');
253          archRappi>>pedido;
254          posPedido=buscarPedido(pedido,arrPedidos,cantPedidos);
255          if (posPedido!=NO_ENCONTRADO){
256              arrPedidos[posPedido].hora=hora;
257              arrPedidos[posPedido].repartidorRappi=new char[strlen(repartidor)+1];
258              strcpy(arrPedidos[posPedido].repartidorRappi,repartidor);
259              arrPedidos[posPedido].codUsuario=new char[strlen(codUsuario)+1];
260              strcpy(arrPedidos[posPedido].codUsuario,codUsuario);
261
                 posUsuario=buscarUsuario(codUsuario,arrDirecciones,cantDirecciones);

262              if (posUsuario!=NO_ENCONTRADO){
263                  arrPedidos[posPedido].direccion=new
                     char[strlen(arrDirecciones[posUsuario].direccion)+1];
264
                     strcpy(arrPedidos[posPedido].direccion,arrDirecciones[posUsuario].direcci
                     on);
265                  arrPedidos[posPedido].distrito=new
                     char[strlen(arrDirecciones[posUsuario].distrito)+1];
266
                     strcpy(arrPedidos[posPedido].distrito,arrDirecciones[posUsuario].distrito
                     );
267              }
268          }
```

```
269         }
270     }
271
272     int buscarPedido(int pedido,struct Pedido *arrPedidos,int cantPedidos){
273         for (int i=0;i<cantPedidos;i++)
274             if (arrPedidos[i].codigo==pedido) return i;
275         return NO_ENCONTRADO;
276     }
277
278     void emitirReporte(struct Pedido *arrPedidos,int cantPedidos){
279         ofstream archReporte("ReportePedidos.txt",ios::out);
280         if (not archReporte.is_open()){
281             cout<<"Error al abrir el archivo ReportePedidos.txt"<<endl;
282             exit(1);
283         }
284         archReporte<<"tp_Rappi"<<endl;
285         archReporte<<"REPORTE DE PEDIDOS PARA EL DIA DE HOY"<<endl;
286         archReporte<<setprecision(2)<<fixed;
287         double total=0;
288         for (int i=0;i<cantPedidos;i++){
289             imprimirPedido(arrPedidos[i],archReporte);
290             total+=arrPedidos[i].montoTotal;
291         }
292         imprimirLinea('=',TAM_LINEA,archReporte);
293         archReporte<<"Total Vendido: S/."<<setw(10)<<total<<endl;
294         imprimirLinea('=',TAM_LINEA,archReporte);
295     }
296
297     void imprimirPedido(struct Pedido pedidoDat,ofstream &archReporte){
298         int hh,mm;
299         imprimirLinea('=',TAM_LINEA,archReporte);
300         archReporte<<"Pedido "<<setw(5)<<pedidoDat.codigo<<") Nombre Rappi: ";
301         archReporte<<left<<setw(30)<<pedidoDat.repartidorRappi<<right<<" Hora:";
302         hh=pedidoDat.hora/60;
303         mm=pedidoDat.hora%60;
304         archReporte<<setfill('0')<<setw(2)<<hh<<':'<<setw(2)<<mm<<setfill(' ')<<left<<endl;
305         archReporte<<right<<setw(20)<<"Destino: "<<setw(15)<<left<<pedidoDat.distrito<<",";
306         archReporte<<setw(40)<<pedidoDat.direccion<<endl;
307         archReporte<<right<<setw(20)<<"Usuario:
            "<<setw(10)<<left<<pedidoDat.codUsuario<<endl;
308         archReporte<<right<<"Datos de la boleta:"<<endl;
309         archReporte<<"PRODUCTO"<<setw(44)<<"PRECIO UNITARIO"<<setw(12)<<"CANTIDAD";
310         archReporte<<setw(15)<<"PRECIO TOTAL"<<endl;
311         imprimirLinea('-',TAM_LINEA,archReporte);
312         for (int i=0;i<pedidoDat.cantProductos;i++){
313             double total=pedidoDat.productos[i].preUni*pedidoDat.productos[i].cantidad;
314             archReporte<<left<<setw(35)<<pedidoDat.productos[i].nombre<<right;
315             archReporte<<setw(10)<<pedidoDat.productos[i].preUni;
316             archReporte<<setw(15)<<pedidoDat.productos[i].cantidad;
317             archReporte<<setw(15)<<total<<endl;
318         }
319         imprimirLinea('-',TAM_LINEA,archReporte);
320         archReporte<<"Resumen:"<<setw(52)<<pedidoDat.cantProductos<<" items";
321         archReporte<<setw(9)<<pedidoDat.montoTotal<<endl;
322     }
323
324     void imprimirLinea(char car, int cant, ofstream &archivo){
325         for (int i=0;i<cant;i++)
326             archivo.put(car);
327         archivo<<endl;
328     }
329
330
331
```