```cpp
/*
 * Archivo:   main.cpp
 * Autor: Oscar Dueñas Damian - Oscar DD.
 * Codigo PUCP: 20180146
 * Created on 23 de septiembre de 2020, 11:04 PM
 */

#include <iostream>
#include <iomanip>
#include <fstream>
#include "funciones_aux.h"
using namespace std;

int main(int argc, char** argv) {
    crearRegistroBin();
    crearPrecioBin();
    fstream archReg = abrirArchLE("registro.bin",'B');
    actualizarNC(archReg);
    ofstream archRep = abrirArchE("Reporte.txt",'T');
    imprimirReporte(archReg,archRep);
    archReg.close();
    archRep.close();
    return (EXIT_SUCCESS);
}

```

```
/*
 * Archivo:   funciones_aux.h
 * Author: Oscar Dueñas Damian Oscar DD.
 * Codigo PUCP: 20180146
 * Created on 23 de septiembre de 2020, 11:05 PM
 */

#ifndef FUNCIONES_AUX_H
#define FUNCIONES_AUX_H
#include <fstream>
using namespace std;

void imprimeLinea (char , int , ofstream &);
ifstream abrirArchL (const char *, char );
ofstream abrirArchE (const char *, char );
fstream abrirArchLE (const char *, char );
void crearRegistroBin ();
void guardarDatosReg(int , int , int , int , char , int , int , double ,
        int , ofstream &);
void crearPrecioBin();
void guardarPrecios (int , double , int , int , ofstream &);
double buscarPrecio (double , int , int , ifstream &);
void actualizarNC (fstream &);
void actualizarReg (int , double , fstream &);
void imprimirReporte (fstream &, ofstream &);
void imprimirDatos (int , int , int , int , char *, int ,
        int , double , int , ofstream &)
void imprimirCabecera (ofstream &);

#endif /* FUNCIONES_AUX_H */

```

```cpp
/*
 * Archivo:   funciones_aux.cpp
 * Autor: Oscar Dueñas Damian - Oscar DD.
 * Codigo PUCP: 20180146
 * Created on 23 de septiembre de 2020, 11:05 PM
 */

#include <iostream>
#include <iomanip>
#include <fstream>
#define MAX_CAR 120
using namespace std;

void imprimeLinea (char car, int num, ofstream &arch) {
    for (int i = 0; i < num; i++) arch.put(car);
    arch << endl;
}

ifstream abrirArchL (const char *nombre, char modo) {
    ifstream arch;
    if (modo == 'T') arch.open(nombre, ios::in);
    else arch.open(nombre, ios::in | ios::binary);
    if (!arch) {
        cout << "ERROR: no se pudo abrir el archivo " << nombre << endl;
        exit(1);
    }
    return arch;
}

ofstream abrirArchE (const char *nombre, char modo) {
    ofstream arch;
    if (modo == 'T') arch.open(nombre, ios::out);
    else arch.open(nombre, ios::out | ios::binary);
    if (!arch) {
        cout << "ERROR: no se pudo abrir el archivo " << nombre << endl;
        exit(1);
    }
    return arch;
}

fstream abrirArchLE (const char *nombre, char modo) {
    fstream arch;
    if (modo == 'T') arch.open(nombre, ios::in | ios::out);
    else arch.open(nombre, ios::in | ios::out | ios::binary);
    if (!arch) {
        cout << "ERROR: no se pudo abrir el archivo " << nombre << endl;
        exit(1);
    }
    return arch;
}

void guardarDatosReg(int cliente, int dd, int mm, int aa, char *doc, int serie,
        int sec, double monto, int referencia, ofstream &archRegBin) {
    archRegBin.write(reinterpret_cast<const char*>(&cliente),sizeof(int));
    archRegBin.write(reinterpret_cast<const char*>(&dd),sizeof(int));
    archRegBin.write(reinterpret_cast<const char*>(&mm),sizeof(int));
    archRegBin.write(reinterpret_cast<const char*>(&aa),sizeof(int));
    archRegBin.write(reinterpret_cast<const char*>(doc),sizeof(char)*5);
    archRegBin.write(reinterpret_cast<const char*>(&serie),sizeof(int));
    archRegBin.write(reinterpret_cast<const char*>(&sec),sizeof(int));
    archRegBin.write(reinterpret_cast<const char*>(&monto),sizeof(double));
    archRegBin.write(reinterpret_cast<const char*>(&referencia),sizeof(int));
}

void crearRegistroBin () {
    ifstream archReg = abrirArchL("registro.txt",'T');
    ofstream archRegBin = abrirArchE("registro.bin",'B');
    int cliente, dd, mm, aa, serie, sec, referencia;
    char doc[5], car;
    double monto;
    while (1) {
        archReg >> cliente;
        if (archReg.eof()) break;
```

```
74          archReg >> dd >> car >> mm >> car >> aa >> doc >> serie >> car >> sec
75                  >> monto >> referencia;
76          guardarDatosReg(cliente, dd, mm, aa, doc, serie, sec, monto, referencia,
77                  archRegBin);
78      }
79      archReg.close(); archRegBin.close();
80  }
81
82  void guardarPrecios (int codProd, double precio, int mm, int aa, ofstream &archPBin) {
83      archPBin.write(reinterpret_cast<const char*>(&codProd),sizeof(int));
84      archPBin.write(reinterpret_cast<const char*>(&precio),sizeof(double));
85      archPBin.write(reinterpret_cast<const char*>(&mm),sizeof(int));
86      archPBin.write(reinterpret_cast<const char*>(&aa),sizeof(int));
87  }
88
89  void crearPrecioBin () {
90      ifstream archP = abrirArchL("precios.txt",'T');
91      ofstream archPBin = abrirArchE("precios.bin",'B');
92      int flag = 0, mm, aa, codProd;
93      double precio;
94      char car;
95      while (flag != 1) {
96          if (flag == -1) mm = codProd;
97          else archP >> mm;
98          archP >> car >> aa;
99          flag = 0;
100         while (1) {
101             archP >> codProd;
102             if (archP.eof()) {
103                 flag = 1;
104                 break;
105             }
106             if (codProd <= 12) {
107                 flag = -1;
108                 break;
109             }
110             archP >> precio;
111             guardarPrecios(codProd,precio,mm,aa,archPBin);
112         }
113     }
114     archP.close(); archPBin.close();
115 }
116
117 double buscarPrecio (double producto, int mes, int anio, ifstream &archP) {
118     int codProd, mm, aa;
119     double precio;
120     int tamDelRegistro = sizeof(int)*3 + sizeof(double);
121     archP.seekg(0, ios::end);
122     int tamDelArch = archP.tellg();
123     archP.seekg(0, ios::beg);
124     int numRegistros = tamDelArch/tamDelRegistro;
125     for (int i = 0; i < numRegistros; i++) {
126         archP.seekg(i*tamDelRegistro, ios::beg);
127         archP.read(reinterpret_cast<char*>(&codProd),sizeof(int));
128         archP.read(reinterpret_cast<char*>(&precio),sizeof(double));
129         archP.read(reinterpret_cast<char*>(&mm),sizeof(int));
130         archP.read(reinterpret_cast<char*>(&aa),sizeof(int));
131         if (codProd == producto && mm == mes && aa == anio) return precio;
132     }
133 }
134
135 void actualizarReg (int docDev, double monto_total, fstream &archReg) {
136     int referencia;
137     int tamDelRegistro = sizeof(int)*7 + sizeof(double) + sizeof(char)*5;
138     archReg.seekg(0, ios::end);
139     int tamDelArch = archReg.tellg();
140     archReg.seekg(0, ios::beg);
141     int numRegistros = tamDelArch/tamDelRegistro;
142     for (int i = 0; i < numRegistros; i++) {
143         archReg.seekg(i*tamDelRegistro, ios::beg);
144         archReg.seekg(sizeof(int)*6 + sizeof(char)*5 + sizeof(double), ios::cur);
145         archReg.read(reinterpret_cast<char*>(&referencia),sizeof(int));
146         if (docDev == referencia) {
```

```
147             archReg.seekg(i*tamDelRegistro + sizeof(int)*6 + sizeof(char)*5,
                ios::beg);
148             archReg.write(reinterpret_cast<const char*>(&monto_total),sizeof(double));
149             break;
150         }
151     }
152 }
153
154 void actualizarNC (fstream &archReg) {
155     ifstream archDev = abrirArchL("devolucion.txt",'T');
156     ifstream archP = abrirArchL("precios.bin",'B');
157     int docDev, dd, mm, aa, producto, cantidad, docDevAnt, cont = 0;
158     double precio, total = 0;
159     char car;
160     while (1) {
161         archDev >> docDev;
162         if (archDev.eof()) {
163             actualizarReg(docDevAnt,total,archReg);
164             break;
165         }
166         archDev >> dd >> car >> mm >> car >> aa >> producto >> cantidad;
167         if (cont > 0 && docDev != docDevAnt) {
168             actualizarReg(docDevAnt,total,archReg);
169             total = 0;
170         }
171         precio = buscarPrecio(producto,mm,aa,archP);
172         total += cantidad*precio;
173         cont++;
174         docDevAnt = docDev;
175     }
176     archDev.close();
177     archP.close();
178 }
179
180 void imprimirCabecera (ofstream &archRep) {
181     archRep << "REGISTRO DE VENTAS" << endl;
182     imprimeLinea('=',MAX_CAR,archRep);
183     archRep << "FECHA"
184             << setw(20) << "CLIENTE"
185             << setw(10) << "TIPO"
186             << setw(15) << "DOCUMENTO"
187             << setw(15) << "MONTO"
188             << setw(10) << "IGV"
189             << setw(20) << "MONTO TOTAL"
190             << setw(15) << "REFERENCIA" << endl;
191     imprimeLinea('-',MAX_CAR,archRep);
192 }
193
194 void imprimirDatos (int cliente, int dd, int mm, int aa, char *doc, int serie,
195         int sec, double monto, int referencia, ofstream &archRep) {
196     double igv = monto*0.18;
197     archRep.fill('0');
198     archRep << setw(2) << dd << '/' << setw(2) << mm << '/' << setw(4) << aa;
199     archRep.fill(' ');
200     archRep << setw(15) << cliente
201             << setw(10) << doc;
202     archRep << "       ";
203     archRep.fill('0');
204     archRep << setw(4) << serie << '-' << setw(5) << sec;
205     archRep.fill(' ');
206     archRep << setw(15) << monto
207             << setw(10) << igv
208             << setw(18) << monto+igv
209             << setw(15) << referencia << endl;
210 }
211
212 void imprimirReporte (fstream &archReg, ofstream &archRep) {
213     imprimirCabecera(archRep);
214     archRep.precision(2); archRep << fixed;
215     int cliente, dd, mm, aa, serie, sec, referencia;
216     char doc[5], car;
217     double monto, total = 0;
218     int tamDelRegistro = sizeof(int)*7 + sizeof(double) + sizeof(char)*5;
```

```
219         archReg.seekg(0, ios::end);
220         int tamDelArch = archReg.tellg();
221         archReg.seekg(0, ios::beg);
222         int numRegistros = tamDelArch/tamDelRegistro;
223         for (int i = 0; i < numRegistros; i++) {
224             archReg.seekg(i*tamDelRegistro, ios::beg);
225             archReg.read(reinterpret_cast<char*>(&cliente),sizeof(int));
226             archReg.read(reinterpret_cast<char*>(&dd),sizeof(int));
227             archReg.read(reinterpret_cast<char*>(&mm),sizeof(int));
228             archReg.read(reinterpret_cast<char*>(&aa),sizeof(int));
229             archReg.read(reinterpret_cast<char*>(doc),sizeof(char)*5);
230             archReg.read(reinterpret_cast<char*>(&serie),sizeof(int));
231             archReg.read(reinterpret_cast<char*>(&sec),sizeof(int));
232             archReg.read(reinterpret_cast<char*>(&monto),sizeof(double));
233             archReg.read(reinterpret_cast<char*>(&referencia),sizeof(int));
234             total += monto*1.18;
235             imprimirDatos(cliente,dd,mm,aa,doc,serie,sec,monto,referencia,archRep);
236         }
237         imprimeLinea('=',MAX_CAR,archRep);
238         archRep << "RESUMEN" << endl << "SALDO FINAL: " << total << endl;
239     }
```