Práctica 4
Programación Dinámica

Por: Joaquín Sergio García Ibáñez
Juan Navarro Maldonado

Clase Empresa

```
class Empresa{
private:
    string nombre;
    int acciones;
    double precioPorAccion;
    double beneficioPorAccion;
    double comisionPorOperacion;


public:
    double accionXbeneficioXcomision;
    Empresa(){
        this->nombre = " ";
        this->acciones = 0;
        this->precioPorAccion = 0;
        this->beneficioPorAccion = 0;
        this->comisionPorOperacion = 0;
    }

    Empresa(string nombre,int acciones, double precio, double beneficio, double comision){
        this->nombre = nombre;
        this->acciones = acciones;
        this->precioPorAccion = precio;
        this->beneficioPorAccion = beneficio/100;
        this->comisionPorOperacion = comision / 100;

        this->accionXbeneficioXcomision = ((precio * beneficioPorAccion) - (precio*comisionPorOperacion)) -precio;
    }

    int numeroMaximoAccionesPuedoComprar(int dinero){
        int numAcciones = 0;
        double precioTotal = 0;
        while(numAcciones <= this->acciones and dinero > precioTotal){
            precioTotal = this->precioPorAccion * numAcciones + (this->comisionPorOperacion * numAcciones);
            //cout << "\nprecioTotal: " << precioTotal << "\nNumAcciones "  << numAcciones << endl;
            numAcciones++;


        }

        if(numAcciones != 0){
            numAcciones--;
        }
        if(dinero<precioTotal){
            numAcciones--;
        }

        //cout << "Detalles compra: " << numAcciones << endl;
        return numAcciones;
    }

    void compraAcciones(double &dinero, int numAcciones){
        //cout << numAcciones << endl;
        double precioTotal = 0;
        precioTotal = this->precioPorAccion * numAcciones + (this->comisionPorOperacion * numAcciones);
        dinero = dinero - precioTotal;
        this->acciones = this->acciones-numAcciones;
    }

    double calcularBeneficioTotal(int numAccionesInvertir) {
        double calculo = numAccionesInvertir * this->accionXbeneficioXcomision;
        return calculo;
    }

};
```

Inicialización de los objetos de la clase empresa

```
391  int main(){
392      double dinero = 100;
393      Empresa empresa1("Fruterias Loli", NUM_ACCIONES, 1, 105.0, 10.0);
394      Empresa empresa2("Bar Paco", NUM_A
395      Empresa empresa3("Chuches Paqui", NUM_ACCIONES,10,120.0,5.0);
396      Empresa empresa4("Car-Wash Asuncion", NUM_ACCIONES,5,102.0,10.0);
397      Empresa empresa5("Ferreteria Manolo",NUM_ACCIONES,5,200.0,1.0);
398      Empresa empresa6("Academia Los Patos", NUM_ACCIONES, 1, 300, 10);
399      vector<Empresa> vectorEmpresas;
400      vectorEmpresas.push_back(empresa1);
401      vectorEmpresas.push_back(empresa2);
402      vectorEmpresas.push_back(empresa3);
403      vectorEmpresas.push_back(empresa4);
404      vectorEmpresas.push_back(empresa5);
405      vectorEmpresas.push_back(empresa6);
406
```

```
const int NUM_ACCIONES = 10
```

Algoritmos Básicos

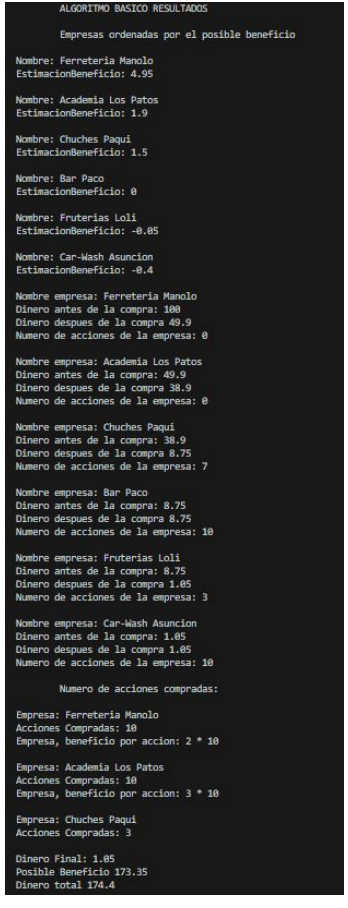
```
133 vector<tuple<Empresa, int>> algoritmoBasico(vector<Empresa> empresas, double &dineroInicial){
134     vector<tuple<Empresa, int>> devolver;
135     set<Empresa> listaEmpresasOrdenada;
136 
137     for(Empresa empresa: empresas){
138         listaEmpresasOrdenada.insert(empresa);
139     }
140 
141     auto iterador = listaEmpresasOrdenada.begin();
142     Empresa empresa1 = *iterador;
143 
144     cout << "\n\tEmpresas ordenadas por el posible beneficio" << endl ;
145     for(auto it = listaEmpresasOrdenada.begin(); it != listaEmpresasOrdenada.end(); it++){
146         Empresa empresa = *it;
147 
148         cout << "\nNombre: " << empresa.getNombre() << "\nEstimacionBeneficio: " << empresa.getaccionXbeneficio
149     }
150 
151 
152     //empresa1.compraAcciones(dineroInicial, 2);
153 
154 
155     for(auto it = listaEmpresasOrdenada.begin(); it != listaEmpresasOrdenada.end(); it++){
156         Empresa empresa = *it;
157         cout << "\nNombre empresa: " << empresa.getNombre() << endl;
158         int numAcciones = empresa.numeroMaximoAccionesPuedoComprar(dineroInicial);
159         cout << "Dinero antes de la compra: " << dineroInicial << endl;
160         empresa.compraAcciones(dineroInicial, numAcciones);
161         cout << "Dinero despues de la compra " << dineroInicial << endl;
162         cout << "Numero de acciones de la empresa: " << empresa.getAcciones() << endl;
163         devolver.push_back(make_tuple(empresa, numAcciones));
164     }
165 
166 
167 
168     return devolver;
169 }
170 
```

```
185 vector<tuple<Empresa, int>> algoritmoBasico2(vector<Empresa> empresas, double &dineroInicial){
186     vector<tuple<Empresa, int>> devolver;
187     set<Empresa> listaEmpresasOrdenada;
188 
189     for(Empresa empresa: empresas){
190         listaEmpresasOrdenada.insert(empresa);
191     }
192 
193     auto iterador = listaEmpresasOrdenada.begin();
194     Empresa empresa1 = *iterador;
195 
196     cout << "\n\tEmpresas ordenadas por el posible beneficio" << endl ;
197     for(auto it = listaEmpresasOrdenada.begin(); it != listaEmpresasOrdenada.end(); it++){
198         Empresa empresa = *it;
199 
200         cout << "\nNombre: " << empresa.getNombre() << "\nEstimacionBeneficio: " << empresa.getaccionXbeneficio
201     }
202 
203 
204     //empresa1.compraAcciones(dineroInicial, 2);
205 
206 
207     for(auto it = listaEmpresasOrdenada.begin(); it != listaEmpresasOrdenada.end(); it++){
208         Empresa empresa = *it;
209 
210         if(empresa.getaccionXbeneficioXcomision() > 0){
211             cout << "\nNombre empresa: " << empresa.getNombre() << endl;
212             int numAcciones = empresa.numeroMaximoAccionesPuedoComprar(dineroInicial);
213             cout << "Dinero antes de la compra: " << dineroInicial << endl;
214             empresa.compraAcciones(dineroInicial, numAcciones);
215             cout << "Dinero despues de la compra " << dineroInicial << endl;
216             cout << "Numero de acciones de la empresa restantes: " << empresa.getAcciones() << endl;
217             devolver.push_back(make_tuple(empresa, numAcciones));
218         }
219         else{
220             //cout << "\nDe esta empresa no se compra: " <<  empresa.getaccionXbeneficioXcomision() << endl;
221         }
222     }
223     return devolver;
224 }
```

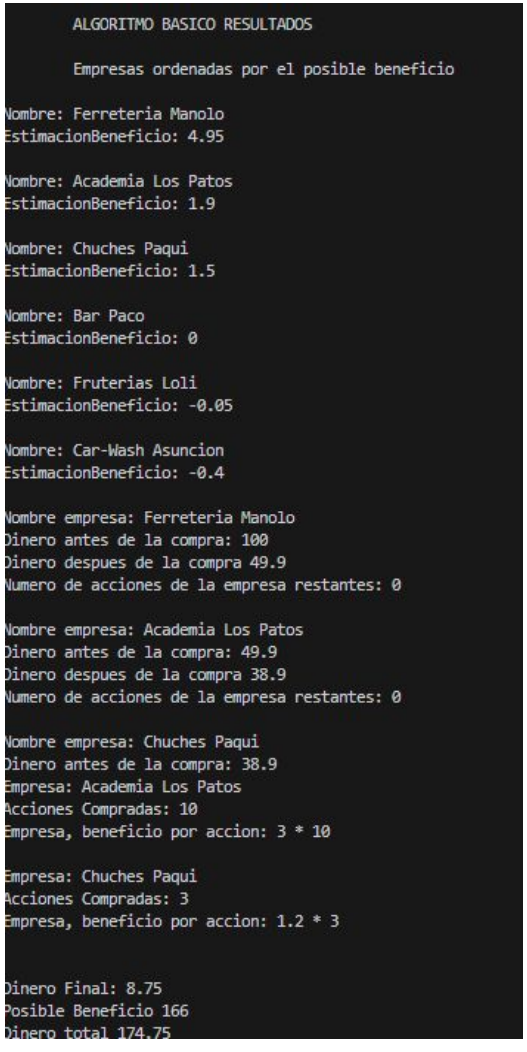
SOBRECARGA DEL OPERADOR <

```
114   bool operator<(const Empresa& a, const Empresa& n) {
115       return a.accionXbeneficioXcomision > n.accionXbeneficioXcomision;
116   }
```

Ejecución Algoritmos Básicos



VS



Algoritmo Programación Dinámica pt1

```
265  vector<tuple<Empresa, int>> ideaOriginal2(std::vector<Empresa>& empresas, double& capital) {
266      int numEmpresas = empresas.size();
267      double beneficios[empresas.size()][NUM_ACCIONES];
268      vector<tuple<Empresa, int>> devolver;
269      vector<Empresa> empre;
270      int max = 0;
271
272
273      //Caso base
274      for (int i = 0; i < numEmpresas; i++){
275          beneficios[i][0] = 0;
276      }
277
278      for(int i = 0; i < numEmpresas; i++){
279          for(int j = 1; j <= NUM_ACCIONES; j++){
280              beneficios[i][j] = empresas[i].getaccionXbeneficioXcomision() * j;
281          }
282      }
283
```

Algoritmo Programación Dinámica pt2

```
303     //Paso 2
304     while(capital_aux > 0 and puedeComprar){
305         //cout << "\nIteracion " << ++contador << endl;
306         max = 0;
307         for(int j = 1; j <= NUM_ACCIONES; j++){
308             for(int i = 0; i < numEmpresas; i++){
309                 if(beneficios[i][j] > max and !findOnTuple(listaCompra, i, j, empresas)){
310                     //cout << "Mejor empresa " << empresas[i].getNombre() << endl;
311                     max = beneficios[i][j];
312                     aux_i = i;
313                     aux_j = j;
314                 }
315             }
316             if(max != 0){
317                 capital_aux -= empresas[aux_i].getPrecio()+empresas[aux_i].getComision();
318                 //cout << "Capitalaux: " << capital_aux << endl;
319                 if(capital_aux < 0){
320                     //cout << "Se acabo el dinero"<< endl;
321                     puedeComprar = false;
322 
323                 }
324                 //cout << "Compramos acciones de: " << empresas[aux_i].getNombre() << " Cantidad: " << aux_j << endl;
325                 else
326                 listaCompra.push_back(make_tuple(empresas[aux_i], aux_j));
327 
328 
329             }
330 
331         }
332     }
```

Algoritmo Programación Dinámica pt3

```
344     for(const auto &it: listaCompra){
345         //cout << "Devolver.size() = " << devolver.size() << endl;
346         Empresa elemento1 = get<0>(it);
347         int elemento2 = get<1>(it);
348         //cout << "\nNombre empresa: " << elemento1.getNombre() << " == " << empresaAux.getNombre() << endl;
349 
350         if(empresaAux.getNombre() == elemento1.getNombre()){
351             //cout << "if " << elemento2 << " > " << numAccionesAux << endl;
352             if(numAccionesAux<elemento2){
353                 numAccionesAux = elemento2;
354             }
355         }else{
356             if(primeraIteracion != 0)
357                 devolver.push_back(make_tuple(empresaAux, numAccionesAux));
358 
359             empresaAux = elemento1;
360             numAccionesAux = elemento2;
361             primeraIteracion++;
362         }
363     }
364 
```

Ejecución Algoritmo Programación Dinámica

```
        ALGORITMOS DE PROGRAMACION DINAMICA RESULTADOS
Matriz beneficios
Fruterias Loli  -0.05 -0.1 -0.15 -0.2 -0.25 -0.3 -0.35 -0.4 -0.45 -0.5
Bar Paco        0 0 0 0 0 0 0 0 0 0
Chuches Paqui   1.5 3 4.5 6 7.5 9 10.5 12 13.5 15
Car-Wash Asuncion       -0.4 -0.8 -1.2 -1.6 -2 -2.4 -2.8 -3.2 -3.6 -4
Ferreteria Manolo       4.95 9.9 14.85 19.8 24.75 29.7 34.65 39.6 44.55 49.5
Academia Los Patos      1.9 3.8 5.7 7.6 9.5 11.4 13.3 15.2 17.1 19

Vector devolver

Nombre empresa: Ferreteria Manolo Acciones compradas: 10
Dinero antes de la compra: 100
Dinero despues de la compra 49.9
Numero de acciones de la empresa restantes: 0

Nombre empresa: Academia Los Patos Acciones compradas: 10
Dinero antes de la compra: 49.9
Dinero despues de la compra 38.9
Numero de acciones de la empresa restantes: 0

Nombre empresa: Chuches Paqui Acciones compradas: 3
Dinero antes de la compra: 38.9
Dinero despues de la compra 8.75
Numero de acciones de la empresa restantes: 7

Empresa: Ferreteria Manolo
Acciones Compradas: 10
Empresa, beneficio por accion: 2 * 10

Empresa: Academia Los Patos
Acciones Compradas: 10
Empresa, beneficio por accion: 3 * 10

Empresa: Chuches Paqui
Acciones Compradas: 3
Empresa, beneficio por accion: 1.2 * 3


Dinero final: 8.75
Posible Beneficio 166
Dinero total 174.75
```