Universidad Blas Pascal

Materia: Algoritmos y Estructuras de Datos I

Docente: Gutiérrez, Julio Marcelo

Integrante: Abril Perrig, David.

2022

Introducción:

Este informe tiene el objetivo de Presentar el diseño e implementación del código en C++ que responde a la consigna del segundo parcial o trabajo practico, para ello primero incluiré la consigna.

Consigna:

Una empresa realiza el diseño y desarrollo de un nuevo proceso de fabricación de materiales. Antes de ponerlo en producción, se deciden realizar varias pruebas para verificar el funcionamiento del mismo. Las pruebas consisten en llevar a cabo producciones de distintos materiales, los cuales se obtienen mediante la combinación de diferentes componentes. Durante el desarrollo de las pruebas, se realizan un conjunto de mediciones, las cuales se registran en un archivo de texto con el siguiente formato:

Material	id Componente	Medición
x	150	15
x	250	50
z	300	60
z	250	36
…	…	…

Cada material se fabrica en base a un conjunto de componentes. Una muestra de la lista de componentes es la siguiente:

id Componente	Cantidad
150	20
250	50
300	65
….	….

Cada componente se identifica con id, y el dato Cantidad determina la proporción que debe tener cada material de ese componente. Para poder verificar el proceso de fabricación, es necesario desarrollar un análisis del mismo. El análisis consiste en determinar si para cada material se respetan las proporciones correspondientes de cada componente, teniendo en cuenta un nivel de tolerancia de +-5%. Se pide realizar un programa que permita:

● Cargar los datos de las mediciones realizadas en una estructura de datos adecuada para realizar el análisis correspondiente.

● Evaluar para cada medición, si los componentes incorporados en cada material respetan las proporciones indicadas (Considerando la tolerancia definida)

● El programa debe devolver un archivo indicando las correcciones que se deben realizar en cada componente.

● Generar un archivo ordenado por material, conteniendo el total de componentes que necesita para su fabricación.

Nota: Realizar implementación utilizando estructuras de datos revisadas en clase. Implementar con objetos. Considerar que el programa debe realizar el control de N pruebas del proceso

Diseño

Para diseñar este código empecé por pensar como guardar los datos de cada una de las pruebas de la manera con mejor optimización posible. Para ello se me ocurrió crear una cola de objetos y que cada uno de estos objetos de la cola tenga tres atributos, obviamente el material, el id de componente y la medición. Así podía guardar los datos de cada prueba rápidamente. Para este primer sistema también cree una clase llamada “CargaCola”. Luego Cargue mediante código en una matriz los valores de la cantidad de cada componente con su id de componente, esto lo hice ya que el tiempo de carga es muy bajo y por lo tanto mejora la optimización del código. Luego cree otra clase llamada “EvaluarRes” la cual compara los valores de las mediciones en la cola con los que deberían ser, cargados en la matriz y realiza un archivo indicando si la medición es correcta o incorrecta y de cuanto es la corrección en caso de ser

incorrecta. Por último, realice una última clase que utiliza los valores de las mediciones cargadas en la cola para sumar la cantidad de cada material necesaria y los totales los guarda en otro archivo. Obviamente Luego se implementa todo en el main() del programa.

Código

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class prueba
{
private:
    string material;
    int id_componente;
    int medicion;

public:
    prueba()
    {
        material = "a";
        id_componente = 0;
        medicion = 0;
    }
    void setMaterial(string m) { material = m; }
    string getmaterial() { return material; }
    void setid(int id) { id_componente = id; }
    int getid() { return id_componente; }
    void setmedicion(int med) { medicion = med; }
    int getmedicion() { return medicion; }
};
class nodo
{
private:
    prueba dato;
    nodo *puntero;

public:
    nodo() { puntero = NULL; }
    void setdato(prueba d) { dato = d; }
    prueba getdato() { return dato; }
    void setpuntero(nodo *p) { puntero = p; }
    nodo *getpuntero() { return puntero; }
};

class Cola
{
private:
    nodo *raiz;
```

```
44
45 public:
46     Cola()
47     {
48         raiz = NULL;
49     }
50     void insertar(prueba *Dato);
51     prueba sacar();
52 };
53
54 void Cola::insertar(prueba *Dato)
55 {
56     nodo *nuevo = new nodo;
57     nuevo->setdato(*Dato);
58     nuevo->setpuntero(NULL);
59
60     if (raiz == NULL)
61     {
62         raiz = nuevo;
63     }
64     else
65     {
66         nodo *recor = raiz;
67         while (recor->getpuntero() != NULL)
68         {
69             recor = recor->getpuntero();
70         }
71         recor->setpuntero(nuevo);
72     }
73 }
74
75 prueba Cola::sacar()
76 {
77     prueba aux;
78     if (raiz == NULL)
79     {
80         cout << "La cola esta vacia";
81         prueba nuevo;
82         return nuevo;
83     }
84     else
85     {
86         aux = raiz->getdato();
87         nodo *temp = raiz;
88         raiz = raiz->getpuntero();
89         delete temp;
90     }
91     return aux;
92 }
93
94 class CargaCola : public Cola
95 {
```

```
96 public:
97     CargaCola() {}
98     Cola ColaCargada()
99     {
100         string Dato1, Basura;
101         ifstream ArchLista("datostp1.txt");
102         Cola C;
103         int Dato;
104         ArchLista >> Basura >> Basura >> Basura;
105         ArchLista >> Dato1;
106 
107         while (!ArchLista.eof())
108         {
109             prueba *nuevo1 = new prueba;
110             nuevo1->setMaterial(Dato1);
111             ArchLista >> Dato1;
112             Dato = stoi(Dato1);
113             nuevo1->setid(Dato);
114             ArchLista >> Dato1;
115             Dato = stoi(Dato1);
116             nuevo1->setmedicion(Dato);
117             C.insertar(nuevo1);
118             ArchLista >> Dato1;
119         }
120         return C;
121     }
122 };
123 const int compCant[3][2] = {{150, 20}, {250, 50}, {300, 65}};
124 class EvaluarRes
125 {
126 private:
127     Cola C;
128     prueba A;
129 
130 public:
131     EvaluarRes(Cola C1, prueba A1)
132     {
133         C = C1;
134         A = A1;
135     }
136     void Evaluacion()
137     {
138         Cola Extra;
139         ofstream Arch("Evaluaciones.txt");
140         A = C.sacar();
141         prueba *nueva = new prueba;
142         *nueva = A;
143         Extra.insertar(nueva);
144         float correccion;
145         while (A.getid() != 0)
146         {
147 
```

```
148             Arch << A.getmaterial() << " " << A.getid() << " " <<
149 A.getmedicion();
150             for (int i = 0; i < 3; i++)
151             {
152                 if (A.getid() == compCant[i][0])
153                 {
154                     float comp = compCant[i][1];
155                     if (((comp / 100) * 95) > A.getmedicion())
156                     {
157                         Arch << " Medicion Incorrecta ";
158                         correccion = comp - A.getmedicion();
159                         Arch << "+" << correccion;
160                     }
161                     else if (A.getmedicion() > ((comp / 100) * 105))
162                     {
163                         Arch << " Medicion Incorrecta";
164                         correccion = comp - A.getmedicion();
165                         Arch << correccion;
166                     }
167                     else
168                     {
169                         Arch << " Medicion Correcta";
170                     }
171                 }
172             }
173             A = C.sacar();
174             prueba *nueva = new prueba;
175             *nueva = A;
176             Extra.insertar(nueva);
177             Arch << endl;
178         }
179         C = Extra;
180     }
181 };
182
183 class TotalM
184 {
185 private:
186     Cola C;
187     prueba A;
188
189 public:
190     TotalM(Cola C1, prueba A1)
191     {
192         C = C1;
193         A = A1;
194     }
195     void Suma()
196     {
197         ofstream Arch1("Total.txt");
198         Cola Extra;
199         int TotalX=0, TotalY=0, TotalZ=0;
```

```
200
201         A = C.sacar();
202         prueba *nueva = new prueba;
203         *nueva = A;
204         Extra.insertar(nueva);
205
206         while (A.getid() != 0){
207             if(A.getmaterial()=="x")TotalX+=A.getmedicion();
208             else if(A.getmaterial()=="y")TotalY+=A.getmedicion();
209             else if(A.getmaterial()=="z")TotalZ+=A.getmedicion();
210             A = C.sacar();
211             prueba *nueva = new prueba;
212             *nueva = A;
213             Extra.insertar(nueva);
214         }
215         Arch1 << "Total X: "<<TotalX<<endl<< "Total Y: "<<TotalY<<endl<<
216 "Total Z: "<<TotalZ<<endl;
217         C=Extra;
218     }
219 };
220
221 int main()
222 {
223     CargaCola T1;
224     Cola T = T1.ColaCargada();
225     prueba A;
226     EvaluarRes C(T, A);
227     C.Evaluacion();
228     TotalM M(T, A);
        M.Suma();
    }
```