

Programación Orientada a Objetos

Contenedores en C++

Esp. Ing. César Aranda

cesar.aranda@ingenieria.uncuyo.edu.ar

Ingeniería en Mecatrónica

Objetivos y contenidos

- Analizar diferentes tipos de clases que definen a conjuntos de datos y objetos agregados
- Conocer las operaciones asociadas a la gestión de colecciones de objetos
- Discutir aspectos básicos de implementación usando colecciones en lenguaje C++

Clases de colección en C++

- Array
- Vector
- List
- Stack
- Queue
- Set
- Map
-

Contenedores =
Colecciones de Objetos

Contenedores en C++. Operaciones

Array
Vector
List
Stack
Queue
Set
Map

O
p
e
r
a
c
i
o
n
e
s

- Declaración
- Creación
- Añadir un ítem
- Eliminar un ítem
- Recuperar un ítem
- Recorrido del conjunto
- Búsqueda de un ítem
- Transformación
- Copiado (clonado)
- ...

El nombre y el comportamiento de cada operación varía en función de las particularidades del tipo de colección analizada.

Array de Objetos

```
1 #include <sstream>
2 using namespace std;
3
4 #include "Punto.h"
5
6 Punto::Punto(string id, int x, int y) {
7     this->id = id;
8     this->x = x;
9     this->y = y;
10 }
11
12 int Punto::getX() const { return x; }
13
14 int Punto::getY() const { return y; }
15
16 string Punto::getId() const { return id; }
17
18 string Punto::toString() const {
19     stringstream ss;
20     ss << "Punto " << id << "(" << x << ", " << y << ")";
21     return ss.str();
22 }
```

```
6 class Punto {
7 public:
8     Punto(string id, int x, int y);
9
10     int getX() const;
11     int getY() const;
12     string getId() const;
13
14 private:
15     string id;
16     int x;
17     int y;
18 };
19
```

5

```
1 #include <iostream>
2 #include <sstream>
3 #include <string>
4 #include <cstdlib>
5 #include <ctime>
6 using namespace std;
7 #include "Punto.h"
8
9 int main() {
10     //Objeto Punto unico
11     Punto *p = new Punto("A", 10, 15);
12     cout << p->toString() << endl << endl;
13
14     //Array de Objetos Punto con coordenadas al azar
15     Punto *arre[5];
16     srand(time(0));
17     for (int i = 0; i < 5; i++) {
18         ostringstream aux;
19         aux << (char) (65 + i);
20         arre[i] = new Punto(aux.str(), rand() % 100, rand() % 100);
21     }
22
23     for (int i = 0; i < 5; i++) {
24         cout << arre[i]->toString() << endl;
25     }
26
27     return 0;
28 }
```

```
Punto A(10, 15)
Punto A(33, 43)
Punto B(62, 29)
Punto C(0, 8)
Punto D(52, 56)
Punto E(56, 19)
RUN SUCCESSFUL
```

6

Vector de Objetos Punto (1)

```
8 #include <vector>
9 using namespace std;
10 #include "Punto.h"
11
12 int main() {
13     //Colección de Objetos Punto
14     vector <Punto> pts;
15     cout << "Nro de Elementos Inicial: " << pts.size() << endl;
16     cout << "Contenido:" << endl;
17     for (size_t i = 0; i < pts.size(); i++) {
18         cout << pts[i].toString() << endl;
19     }
20     srand(time(0));
21     for (size_t i = 0; i < 5; i++) {
22         ostream aux;
23         aux << (char) (65 + i);
24         Punto p(aux.str(), rand() % 100, rand() % 100);
25         pts.push_back(p);
26     }
27     cout << "Nro Elementos post llenado: " << pts.size() << endl;
28     cout << "Contenido:" << endl;
29     for (size_t i = 0; i < pts.size(); i++) {
30         cout << pts[i].toString() << endl;
31     }
32     return 0;
33 }
```

```
6 class Punto {
7 public:
8     Punto();
9     Punto(string id, int x, int y);
```

```
Nro de Elementos Inicial: 0
Contenido:
Nro Elementos post llenado: 5
Contenido:
Punto A(63, 68)
Punto B(54, 29)
Punto C(9, 3)
Punto D(56, 10)
Punto E(34, 37)
```

7

Vector de Objetos Punto (2)

```
33 if(!pts.empty()){
34     ostream aux;
35     aux << (char) (80);
36     Punto p(aux.str(), rand() % 100, rand() % 100);
37     pts.insert(pts.begin()+3, p);
38 }
39 cout << "Nro Elementos post insercion: " << pts.size() << endl;
40 cout << "Contenido:" << endl;
41 for (size_t i = 0; i < pts.size(); i++) {
42     cout << pts[i].toString() << endl;
43 }
```

```
Nro Elementos post insercion: 6
Contenido:
Punto A(12, 86)
Punto B(84, 7)
Punto C(2, 17)
Punto P(63, 12)
Punto D(34, 53)
Punto E(75, 0)
```

Lista de Objetos Punto

```
10 #include <list>
11 using namespace std;
12 #include "Punto.h"
13
14 int main() {
15     //Colección de Objetos Punto
16     list<Punto> pts;
17     list<Punto>::iterator iter;
18
19     if (pts.empty()) {
20         cout << "La lista está vacía!" << endl;
21     }
22     srand(time(0));
23     for (int i = 0; i < 5; i++) {
24         ostream aux;
25         aux << (char) (65 + i);
26         //Punto p(aux.str(), rand() % 100, rand() % 100);
27         pts.insert(pts.end(), Punto(aux.str(), rand() % 100, rand() % 100));
28     }
29     if (!pts.empty()) {
30         cout << "La lista tiene " << pts.size() << " elementos" << endl;
31     }
32     cout << "El primer elemento es: " << pts.begin()->toString() << endl;
33     cout << "El ultimo elemento es: " << pts.rbegin()->toString() << endl;
34     cout << "O bien todos son:" << endl;
35     for (iter = pts.begin(); iter != pts.end(); iter++)
36         cout << iter->toString() << endl;
37     return 0;
38 }
```

La lista está vacía!
La lista tiene 5 elementos
El primer elemento es: Punto A(69, 97)
El ultimo elemento es: Punto E(21, 76)
O bien todos son:
Punto A(69, 97)
Punto B(49, 62)
Punto C(15, 50)
Punto D(15, 9)
Punto E(21, 76)

9

Cola de Objetos Punto

```
15 //Colección de Objetos Punto
16 queue<Punto> pts;
17
18 if (pts.empty()) {
19     cout << "La cola está vacía!" << endl;
20 }
21 srand(time(0));
22 cout << "Se insertan 3 objetos (A, B y C)!" << endl;
23 pts.push(Punto("A", rand() % 100, rand() % 100));
24 pts.push(Punto("B", rand() % 100, rand() % 100));
25 pts.push(Punto("C", rand() % 100, rand() % 100));
26 cout << "Se muestran y retiran 2 objetos!" << endl;
27 cout << pts.front().toString(); // muestra
28 pts.pop(); // quita
29 cout << pts.front().toString() << endl;
30 pts.pop();
31 cout << "Se insertan 2 objetos (E y H)!" << endl;
32 pts.push(Punto("E", rand() % 100, rand() % 100));
33 pts.push(Punto("H", rand() % 100, rand() % 100));
34 cout << "Se retira un objeto!" << endl;
35 pts.pop();
36 cout << "Se muestra y retira 1 objeto!" << endl;
37 cout << pts.front().toString() << endl;
38 pts.pop();
39 if (!pts.empty()) {
40     cout << "A lista le quedan " << pts.size() << " elemento/s" << endl;
41 }
```

La cola está vacía!
Se insertan 3 objetos (A, B y C)!
Se muestran y retiran 2 objetos!
Punto A(8, 27) Punto B(33, 6)
Se insertan 2 objetos (E y H)!
Se retira un objeto!
Se muestra y retira 1 objeto!
Punto E(66, 10)
A lista le quedan 1 elementos

10

Pila de Objetos Punto

```

15 //Colección de Objetos Punto
16 stack <Punto> pts;
17 if (pts.empty()) {
18     cout << "La pila está vacía!" << endl;
19 }
20 srand(time(0));
21 cout << "Se insertan 5 objetos!" << endl;
22 for (int i = 0; i < 5; i++) {
23     ostream aux;
24     aux << (char) (65 + i);
25     pts.push(Punto(aux.str(), rand() % 100, rand() % 100));
26 }
27
28 if (!pts.empty()) {
29     cout << "La pila tiene ahora " << pts.size() << " elemento/s" << endl;
30 }
31
32 cout << "Se muestran y quitan 2 objetos!" << endl;
33 for (int i = 0; i < 2; ++i) {
34     cout << pts.top().toString() << endl;
35     pts.pop();
36 }
37 cout << "La pila tiene ahora " << pts.size() << " elemento/s" << endl;

```

La pila está vacía!
 Se insertan 5 objetos!
 La pila tiene ahora 5 elemento/s
 Se muestran y quitan 2 objetos!
 Punto E(87, 30)
 Punto D(17, 21)
 La pila tiene ahora 3 elemento/s

11

Ejemplo de Array de Datos Primitivos

```

14 // Declaro un puntero a un doble puntero a char.
15 char **A;
16
17 // Asigno a ese puntero un arreglo de punteros a char.
18 A = new char*[cantFilas];
19
20 // A cada puntero a char del arreglo de doble punteros
21 // le asigno un nuevo arreglo de char.
22 for( int i=0; i<cantFilas; i++ ) {
23     A[i] = new char[cantColumnas];
24 }
25
26 llenar( A, cantFilas, cantColumnas);
27
28 mostrar( A, cantFilas, cantColumnas);
29
30 // Libero la memoria dinámica asignada.
31 for( int i=0; i<3; i++) {
32     delete[] A[i];
33 }
34 delete[] A;
35
36 return 0;
37
38 }
39
40 void llenar( char** A, int cantFilas, int cantColumnas ) {
41     for( int i=0; i<cantFilas; i++ )
42         for( int j=0; j<cantColumnas; j++ )
43             A[i][j] = rand()%6+31;
44 }

```

Ing. César Aranda

12

Encapsular la Matriz en una Clase

```

4  template <typename MM>
5  class MatrizDinamica
6  {
7  private:
8      MM** matriz;
9      int filas;
10     int columnas;
11
12 public:
13     MatrizDinamica(int filas, int columnas) throw(runtime_error) {
14         if (filas <= 0 || columnas <= 0) {
15             throw runtime_error("Cantidad de filas y columnas deben ser mayor a 0");
16         }
17         this->filas = filas;
18         this->columnas = columnas;
19         matriz = new MM*[filas];
20         for (int i = 0; i < filas; i++) {
21             matriz[i] = new MM[columnas];
22         }
23     }
24
25     ~MatrizDinamica() {
26         for (int i = 0; i < this->filas; i++) {
27             delete [] matriz[i];
28         }
29         delete [] matriz;
30     }
31
32     MM getValor(int fila, int columna) throw(runtime_error) {
33         if (fila < 0 || fila >= this->filas) {
34             throw runtime_error("La fila no es valida.");
35         }
36     }

```

13

Ejemplo de Uso

```

1  #include <iostream>
2  using namespace std;
3  #include "MatrizDinamica.cpp"
4
5  /* Ejemplo de uso de la clase.
6   Primer ciclo: llena la matriz
7   Segundo ciclo: lo muestra en pantalla */
8
9  int main(){
10
11     MatrizDinamica<int> matriz(10, 7);
12
13     for (int i = 0; i < matriz.getFilas(); i++) {
14         for (int j = 0; j < matriz.getColumnas(); j++) {
15             matriz.setValor(i, j, i*j);
16         }
17     }
18
19     for (int i = 0; i < matriz.getFilas(); i++) {
20         for (int j = 0; j < matriz.getColumnas(); j++) {
21             cout << matriz.getValor(i, j) << "\t";
22         }
23         cout << endl << endl;
24     }
25
26     return 0;
27 }

```

0	0	0	0	0	0	0
0	1	2	3	4	5	6
0	2	4	6	8	10	12
0	3	6	9	12	15	18
0	4	8	12	16	20	24
0	5	10	15	20	25	30
0	6	12	18	24	30	36
0	7	14	21	28	35	42
0	8	16	24	32	40	48
0	9	18	27	36	45	54

Práctica usando C++

Objetivo:

Aplicar conceptos de colecciones en POO usando lenguaje C++

Consigna:

Modificar el programa que permite gestionar los móviles anterior de modo de modo tal que:

- Se puedan crear N móviles autónomos (en lugar de sólo 3)
- Los móviles se almacenen en una lista de objetos.
- La secuencia de órdenes recibidas se almacene en otra lista adecuada.

Referencias C++

- DEITEL, H. M. y DEITEL, P. J. (2009): Cómo Programar en C/C++, 6ª edición, Prentice-Hall
- BRONSON, Gary (2007): C++ para Ingeniería y Ciencias, 2da edición. Cengage Learning Editores S.A., México. Disponible en <http://www.elsolucionario.org/c-para-ingenieria-y-ciencias-gary-j-bronson-2ed-2/>
- JOYANES AGUILAR, L. y SÁNCHEZ GARCÍA, L. (2006): Programación en C++. Un enfoque práctico. McGraw Hill/Interamericana. Madrid. Disponible en https://www.academia.edu/34123961/Programaci%C3%B3n_en_C_Luis_Joyanes_Aguilar_FREELIBROS
- JOYANES AGUILAR, L (1999): Programación Orientada a Objetos, Prentice-Hall
- http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol1/ch10s03.html
- <http://www.cplusplus.com/reference/>