

Almacenar objetos en un *array*

array es una de las estructuras de datos pertenecientes a la librería STL de C++.

Los *array* son un tipo de arreglo, que almacena diversos tipos de datos (primitivos u objetos), de forma continua, por lo que el acceso a los datos es muy rápido y sencillo. Al igual que un arreglo primitivo, un *array* tiene un tamaño fijo, por lo que no puede aumentar o disminuir su tamaño de manera dinámica.

Para crear un *array* existen variedad de formas. La siguiente es la sintaxis para crear un *array* de tamaño *n*:

```
array<tipoDato, tamaño> nombreArray;
```

Siguiendo con el ejemplo de la clase *Estudiante*, la declaración de un *array* de tamaño 5 que almacene objetos sería así:

```
array <Estudiante, 5> arrayEstudiantes;
```

Asignación en un *array* de objetos (1)

Para asignar (almacenar) objetos en un *array* de tipo clase, se deben seguir los siguientes pasos:

- 1) Declarar el *array* de objetos de la clase
- 2) Crear un objeto de la clase y asignar los valores a sus atributos
- 3) Por medio del [índice] de *array*, guardar el objeto

Continuando con el ejemplo del *array* de tipo *Estudiante* la asignación de un objeto seria así:

- 1) array <*Estudiante*, 5> arrayEstudiantes;
- 2) *Estudiante* obj;
obj.setCodEstudiantil();
obj.setApellido();
obj.setNombre();
obj.setCodPlan();
obj.setNotas();
- 3) arrayEstudiantes[0] = obj;

Asignación en un *array* de objetos (2)

Generalizando el proceso anterior, podemos usar una estructura repetitiva (*for* o *while*) para asignar (almacenar) *n* objetos en un *array* de tipo clase, así:

```
array <Estudiante, 5> arrayEstudiantes; //crear un array de tipo Estudiante con tamaño de 5
```

```
Estudiante obj; // crear un objeto de la clase
```

```
for(int i = 0; i < arrayEstudiantes.size(); i++){ // size() permite calcular el tamaño del array de forma dinamica
```

```
    obj.setCodEstudiantil();
```

```
    obj.setApellido();
```

```
    obj.setNombre();
```

```
    obj.setCodPlan();
```

```
    obj.setNotas();
```

```
    vecEstudiantes[i] = obj;
```

```
}
```

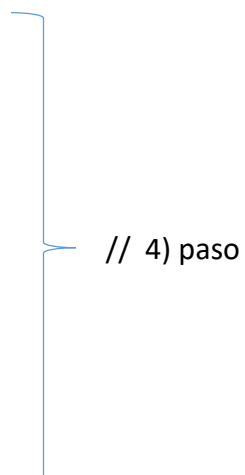
Imprimir un *array* de objetos

Igual como el proceso de asignación (almacenar), para recorrer un *array* se sigue el siguiente proceso:

- 1) Se declara un objeto de tipo de clase que almacena el *array*
- 2) Se usa una estructura repetitiva que se limita hasta el valor que retorne el método *size()*
- 3) Dentro del bucle se asigna al objeto (paso 1) lo que este almacenado en el *array* en el [índice] del bucle
- 4) Con el objeto asignado se llaman a los métodos **get** de la clase

Ejemplo:

```
Estudiante obj;                                // 1) paso
for(int i = 0; i < arrayEstudiantes.size(); i++){ // 2) paso
    cout<<"Datos del Estudiante # "<<(i+1)<<endl;
    obj = arrayEstudiantes[i];                  // 3) paso
    cout<<"Cod Estudiantil:"<< obj.getCodEstudiantil()<<endl;
    cout<<"Apellido:"<< obj.getApellido()<<endl;
    cout<<"Nombre:"<< obj.getNombre()<<endl;
    cout<<"Cod Plan:"<< obj.getCodPlan()<<endl;
    for(int n = 0; n < 3; n++){//recorrer el arreglo de las notas, solo hasta la pos 2. La pos 3 es la definitiva
        cout<<"Nota Parcial "<< (n+1)<<": "<< obj.getNota(n+1)<<endl;
    }
    cout<<"Definitiva: "<< obj.getNota(3)<<endl<<endl;
}
```



Almacenar objetos en un *vector*

vector es una de las estructuras de datos pertenecientes a la librería STL de C++.

Los vectores son un tipo de array, que almacena diversos tipos de datos (primitivos u objetos), de forma continua, por lo que el acceso a los datos es muy rápido y sencillo. Además, un *vector* puede aumentar o disminuir su tamaño de manera dinámica, es decir mientras la aplicación esta ejecutándose, lo cual la hace una de las estructuras favoritas para almacenar y gestionar gran cantidad de datos.

Para crear un *vector* existen variedad de formas, la mas sencilla es la siguiente. Esta crea un *vector* vacío, es decir sin elementos, pero que crece a medida que se añaden valores a éste:

```
vector<tipoDato> nombreVector;
```

Siguiendo con el ejemplo de la clase *Estudiante*, la declaración de un *vector* sin un tamaño inicial que almacene objetos seria así:

```
vector <Estudiante> vecEstudiantes;
```

Asignación en un *vector* de objetos (1)

Para asignar (almacenar) objetos en un *vector* de tipo clase, se deben seguir los siguientes pasos:

- 1) Declarar el *vector* de objetos de la clase
- 2) Crear un objeto de la clase y asignar los valores a sus atributos
- 3) Utilizar el método *push_back(tipoDato)* de *vector* para añadir el objeto al *vector*

Ejemplo con la clase *Estudiante*:

1) *vector* <*Estudiante*> vecEstudiantes;

3) vecEstudiantes.push_back(obj);

2) *Estudiante* obj;

obj.setCodEstudiantil();

obj.setApellido();

obj.setNombre();

obj.setCodPlan();

obj.setNotas();

Asignación en un *vector* de objetos (2)

Generalizando el proceso anterior, podemos usar una estructura repetitiva (*for* o *while*) para asignar (almacenar) *n* objetos en un *vector* de tipo clase, así:

```
vector <Estudiante> vecEstudiantes; //crear un vector de tipo Estudiante sin tamaño inicial
Estudiante obj; // crear un objeto de la clase
int n = 3; // suponer que se desea almacenar 3 objetos inicialmente
for(int i = 0; i < n; i++){
    obj.setCodEstudiantil();
    obj.setApellido();
    obj.setNombre();
    obj.setCodPlan();
    obj.setNotas();
    vecEstudiantes.push_back(obj);
}
```

Imprimir un *vector* de objetos

Igual como el proceso de asignación (almacenar), para recorrer un *vector* se sigue el siguiente proceso:

- 1) Se declara un objeto de tipo de clase que almacena el *vector*
- 2) Se usa una estructura repetitiva que se limita hasta el valor que retorne el método *size()*
- 3) Dentro del bucle se usa el método *at(index)* para obtener el objeto que se almaceno en el índice y se asigna el objeto declarado en el paso 2
- 4) Con el objeto asignado se llaman a los métodos **get** de la clase

Ejemplo:

```
Estudiante obj; // 1) paso
for(int i = 0; i < vecEstudiantes.size(); i++){ // 2) paso
    cout<<"Datos del Estudiante # "<<(i+1)<<endl;
    obj = vecEstudiantes.at(i); // 3) paso
    cout<<"Cod Estudiantil:"<< obj.getCodEstudiantil()<<endl;
    cout<<"Apellido:"<< obj.getApellido()<<endl;
    cout<<"Nombre:"<< obj.getNombre()<<endl;
    cout<<"Cod Plan:"<< obj.getCodPlan()<<endl;
    for(int n = 0; n < 3; n++){//recorrer el arreglo de las notas, solo hasta la pos 2. La pos 3 es la definitiva // 4) paso
        cout<<"Nota Parcial "<< (n+1)<<": "<< obj.getNota(n+1)<<endl;
    }
    cout<<"Definitiva: "<< obj.getNota(3)<<endl<<endl;
}
```


Almacenar objetos en un *list*

list es otra de las estructuras de datos pertenecientes a la librería STL de C++.

Las listas también almacenan diversos tipos de datos (primitivos u objetos), de forma continua, y puede aumentar o disminuir su tamaño de manera dinámica.

Para crear un *list* también existen variedad de formas, la mas sencilla es la siguiente. Esta crea un *list* vacío, es decir sin elementos, pero que crece a medida que se añaden valores a éste:

```
list<tipoDato> nombreLista;
```

Con el ejemplo de la clase *Estudiante*, la declaración de un *list* sin un tamaño inicial que almacene objetos seria así:

```
list <Estudiante> listEstudiantes;
```

Asignación en un *list* de objetos (1)

Para asignar (almacenar) objetos en un *list* de tipo clase, se deben seguir los siguientes pasos:

- 1) Declarar el *list* de objetos de la clase
- 2) Crear un objeto de la clase y asignar los valores a sus atributos
- 3) Utilizar el método *push_back(tipoDato)* de *list* para añadir el objeto a la *list*

Ejemplo con la clase *Estudiante*:

- 1) *list* <*Estudiante*> listEstudiantes;
- 2) *Estudiante* obj;
obj.setCodEstudiantil();
obj.setApellido();
obj.setNombre();
obj.setCodPlan();
obj.setNotas();

- 3) listEstudiantes.push_back(obj);

Igual que *vector*



Asignación en un *list* de objetos (2)

Generalizando el proceso anterior, podemos usar una estructura repetitiva (*for* o *while*) para asignar (almacenar) *n* objetos en un *list* de tipo clase, así:

```
list <Estudiante> listEstudiantes; //crear un list de tipo Estudiante sin tamaño inicial
```

```
Estudiante obj; // crear un objeto de la clase
```

```
int n = 3; // suponer que se desea almacenar 3 objetos inicialmente
```

```
for(int i = 0; i < n; i++){
```

```
    obj.setCodEstudiantil();
```

```
    obj.setApellido();
```

```
    obj.setNombre();
```

```
    obj.setCodPlan();
```

```
    obj.setNotas();
```

```
    listEstudiantes.push_back(obj);
```

```
}
```

Imprimir un *list* de objetos

Igual como el proceso de asignación (almacenar), para recorrer un *list* se sigue el siguiente proceso:

- 1) Se declara un objeto de tipo de clase que almacena el *list*
- 2) Se crea un iterador de tipo objeto de clase que almacena el *list*
- 3) Dentro del bucle se usan los métodos *begin()* y *end()* para recorrer la lista, y se asigna al iterador el objeto que esta almacenado en el list
- 4) Con el objeto asignado se llaman a los métodos **get** de la clase

Ejemplo:

```
Estudiante obj; // 1) paso
list <Estudiante> :: iterator it; // 2) paso
for(it = listEstudiantes.begin(); it != listEstudiantes.end(); ++it){ // 3) paso
    cout<<"Datos del Estudiante # "<<(i+1)<<endl;
    obj = *it; // 3) paso
    cout<<"Cod Estudiantil:"<< obj.getCodEstudiantil()<<endl;
    cout<<"Apellido:"<< obj.getApellido()<<endl;
    cout<<"Nombre:"<< obj.getNombre()<<endl;
    cout<<"Cod Plan:"<< obj.getCodPlan()<<endl;
    for(int n = 0; n < 3; n++){//recorrer el arreglo de las notas, solo hasta la pos 2. La pos 3 es la definitiva
        cout<<"Nota Parcial "<< (n+1)<<": "<< obj.getNota(n+1)<<endl;
    }
    cout<<"Definitiva: "<< obj.getNota(3)<<endl<<endl;
}
```

