Programación y Análisis de Algoritmos

Dr. Norberto Alejandro Hernández Leandro

CIMAT-Unidad Monterrey

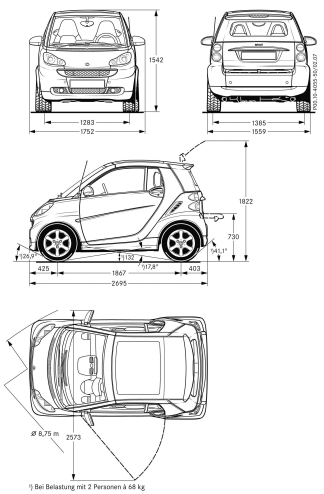
Octubre 2020

Contenido

- Clases
- Templates

Clases

- Las clases son entidades que definen el estado (atributos) y su comportamiento, el cual está definido mediante procedimiento (funciones).
- Así, los objetos representan una instancia de una clase.



Clase

Objeto

Clases

- Las clases son definidas mediante la palabra reservada class

```
1  Class NombreDeClase{
2
3  };
```

- Así mismo, pueden definirse por medio de manera dinámica por medio un procedimiento de construcción y uno de destrucción

```
1  Class NombreDeClase{
2      NombreDeClase();      //Procedimiento que indica
3                            //cómo inicializar
4      ~NombreDeClase();     //Procedimiento que indica
5                            //cómo destruir
6  };
```

Clases

- Representan una abstracción de cualquier objeto
- Permite la modularización del código
- Ofrece trabajar con conceptos como herencia y polimorfismos
- Permite la encapsulación de atributos y procedimiento que tienen la misma naturaleza
- Tanto atributos como procedimientos pueden ser declarados públicos o privados.

```
1  Class NombreDeClase{
2      private:     //Sólo puede ser accedido dentro de la clase
3          int x;
4      public:      //Se puede acceder fuera de la clase
5          NombreDeClase();
6          ~NombreDeClase();
7          void print();
8  };
```

Clases

La definición de los procedimientos de una clase pueden ser definidos directamente dentro de la clase; sin embargo, la mejor práctica se realiza cuando se definen fuera de la clase.

```
1  Class NombreDeClase{
2      private:     //Sólo puede ser accedido dentro de la clase
3      int x;
4      public:      //Se puede acceder fuera de la clase
5      NombreDeClase();
6      ~NombreDeClase();
7      void print();
8  };
9
10 void NombreDeClase::print(){
11     ...
12 }
```

Clases

Al igual que en las estructuras, el acceso a los atributos y procedimientos de una clase se puede realizar por medio de un punto si es un objeto estático; mientras que si es un apuntador, se puede acceder por medio del operador ->.

```
1  int main(){
2      NombreDeClase Objeto1;
3      NombreDeClase *Objeto2;
4      Objeto1.print();
5      Objeto2->print();
6      return 0;
7  }
```

Clases

Herencia

Permite diferentes niveles de abstracción a partir de la declaración de una clase nueva a partir de las ya existentes.

```
1  class Figura {
2      public:
3          int x;
4          int y;
5          virtual void print() = 0;
6  };
7
8  class Circulo : public Figura{
9      private:
10         int radio;
11     public:
12         virtual void print();
13 };
14 void Circulo::print(){
15     ...
16 }
```

Clases

Polimorfismos

Permite que un objeto de un nivel de abstracción más alto pueda tomar la naturaleza de cualquiera de sus subclases.

```
1  class Circulo : public Figura{
2      private:
3          int radio;
4      public:
5          virtual void print();
6  };
7  class Triangulo : public Figura{
8      private:
9          int base;
10         int altura
11     public:
12         virtual void print();
13 };
14
15 int main(){
16     Figura *F1 = new Circulo();
17     Figura *F2 = new Triangulo();
18     return 0;
19 }
```

Templates

- Es una herramienta de C++ que permite escribir código de uso general.
- Se pueden escribir funciones y clases más flexibles, dado que los templates les permiten trabajar con diferentes tipos de datos.
- Permite la reutilización de código.

Templates

- Los templates se pueden definir mediante la palabra reservada template como sigue:

```
1     template <class nombreTemplate>
2     template <typename nombreTemplate>
```

- Su uso, le permite a una función manejar, recibir y/o retornar diferentes tipos de datos.

```
1     template <class numero>
2     numero maximo(numero a, numero b){
3         return (a>b)?a:b;
4     }
5     int main(){
6         int a=1, b=2;
7         double c=1.1, d=1.5;
8         cout << maximo(a,b) << endl;
9         cout << maximo(c,d) << endl;
10    }
```

Templates

- Los templates se pueden definir mediante la palabra reservada template como sigue:

```
1    template <class nombreTemplate>
2    template <typename nombreTemplate>
```

- Su uso, le permite a una función manejar, recibir y/o retornar diferentes tipos de datos.

```
1    template <class numero>
2    numero maximo(numero a, numero b){
3        return (a>b)?a:b;
4    }
5    int main(){
6        int a=1, b=2;
7        double c=1.1, d=1.5;
8        cout << maximo(a,b) << endl;
9        cout << maximo(c,d) << endl;
10   }
```

Precaución: si existe ambigüedad del tipo que debe tomar el template al llamar la función, se tendrá que especificar en la llamada de la función; ej. foo<tipo>(argumentos).

Templates

Los templates también se pueden utilizar en las clases para definir sus atributos.

```
1   template<class numero>
2   class calculadora{
3   private:
4       numero _a;
5       numero _b;
6   public:
7       calculadora(numero a, numero b);
8       print(){
9           cout << "Suma: " << _a+_b << endl;
10          cout << "Resta: " << _a-_b << endl;
11          cout << "Multiplicacion: " << _a*_b << endl;
12          cout << "Division: " << _a/_b << endl;
13      };
14  };
15
16  int main(){
17      calculadora <int> c1(1,2);
18      calculadora <double> c2(1.1,2.2);
19      return 0;
20  }
```