







♠ / Design Patterns / Behavioral patterns / Strategy

Strategy in C++



Back to **Strategy** description

Strategy design pattern demo

Discussion. The Strategy pattern suggests: encapsulating an algorithm in a class hierarchy, having clients of that algorithm hold a pointer to the base class of that hierarchy, and delegating all requests for the algorithm to that "anonymous" contained object.

In this example, the Strategy base class knows how to collect a paragraph of input and implement the skeleton of the "format" algorithm. It defers some details of each individual algorithm to the "justify" member which is supplied by each concrete derived class of Strategy. The Class models an application class that would like to leverage the services of a run-time-specified derived "Strategy" object.

```
#include <iostream.h>
#include <fstream.h>
#include <string.h>
class Strategy;
class TestBed
  public:
    enum StrategyType
        Dummy, Left, Right, Center
    };
    TestBed()
        strategy_ = NULL;
    void setStrategy(int type, int width);
    void doIt();
  private:
    Strategy *strategy_;
};
class Strategy
{
  public:
    Strategy(int width): width_(width){}
    void format()
    {
        char line[80], word[30];
        ifstream inFile("quote.txt", ios::in);
        line[0] = '\0';
        inFile >> word;
        strcat(line, word);
        while (inFile >> word)
        {
            if (strlen(line) + strlen(word) + 1 > width_)
              justify(line);
            else
              strcat(line, " ");
            strcat(line, word);
        }
        justify(line);
    }
  protected:
    int width_;
  private:
```

```
virtual void justify(char *line) = 0;
};
class LeftStrategy: public Strategy
  public:
    LeftStrategy(int width): Strategy(width){}
     /* virtual */void justify(char *line)
    {
        cout << line << endl;</pre>
        line[0] = ' \setminus 0';
    }
};
class RightStrategy: public Strategy
  public:
    RightStrategy(int width): Strategy(width){}
  private:
     /* virtual */void justify(char *line)
    {
        char buf[80];
        int offset = width_ - strlen(line);
        memset(buf, ' ', 80);
        strcpy(&(buf[offset]), line);
        cout << buf << endl;</pre>
        line[0] = '\0';
    }
};
class CenterStrategy: public Strategy
  public:
    CenterStrategy(int width): Strategy(width){}
  private:
     /* virtual */void justify(char *line)
    {
        char buf[80];
        int offset = (width_ - strlen(line)) / 2;
        memset(buf, ' ', 80);
        strcpy(&(buf[offset]), line);
        cout << buf << endl;</pre>
        line[0] = '\0';
    }
};
void TestBed::setStrategy(int type, int width)
```

```
delete strategy_;
  if (type == Left)
    strategy_ = new LeftStrategy(width);
  else if (type == Right)
    strategy_ = new RightStrategy(width);
  else if (type == Center)
    strategy_ = new CenterStrategy(width);
}
void TestBed::doIt()
  strategy_->format();
}
int main()
{
  TestBed test;
  int answer, width;
  cout << "Exit(0) Left(1) Right(2) Center(3): ";</pre>
  cin >> answer;
  while (answer)
    cout << "Width: ";</pre>
    cin >> width;
    test.setStrategy(answer, width);
    test.doIt();
    cout << "Exit(0) Left(1) Right(2) Center(3): ";</pre>
    cin >> answer;
  }
  return 0;
}
```

Output

```
Exit(0) Left(1) Right(2) Center(3): 2
Width: 75
Exit(0) Left(1) Right(2) Center(3): 3
Width: 75
```

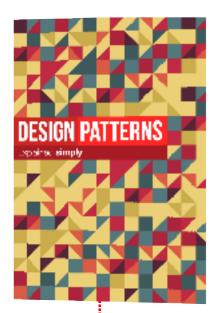
The important lesson we have learned is that development for reuse is complex. If making a good design is difficult, then making a good reusable design is even harder, and any amount of process description cannot substitute for the skill, imagination, and experience of a good designer. A process can only support the creative work, and ensure that things are done and recorded properly.

Read next

This article is taken from our book **Design Patterns Explained Simply**.

All of the design patterns are compiled there. The book is written in clear, simple language that makes it easy to read and understand (just like this article).

We distribute it in PDF & EPUB formats so you can get it onto your iPad, Kindle, or other portable device immediately after your purchase.





Learn more

Code examples

| Java | Strategy in Java |
|--------|--------------------|
| C++ | Strategy in C++ |
| PHP | Strategy in PHP |
| Delphi | Strategy in Delphi |
| | |

Design Patterns My account
AntiPatterns Forum
Refactoring Contact us
UML About us

© 2007-2018 SourceMaking.com All rights reserved.

Terms / Privacy policy