Abstraction

Define `Date::String()` to pass the test in `main()`.

In [ ]:

```
#include <cassert>
#include <string>
#include <vector>

class Date {
 public:
  Date(int day, int month, int year);
  int Day() const { return day_; }
  void Day(int day);
  int Month() const { return month_; }
  void Month(int month);
  int Year() const { return year_; }
  void Year(int year);
  std::string String() const;

 private:
  bool LeapYear(int year) const;
  int DaysInMonth(int month, int year) const;
  int day_{1};
  int month_{1};
  int year_{0};
};

Date::Date(int day, int month, int year) {
  Year(year);
  Month(month);
  Day(day);
}

bool Date::LeapYear(int year) const {
  if (year % 4 != 0)
    return false;
  else if (year % 100 != 0)
    return true;
  else if (year % 400 != 0)
    return false;
  else
    return true;
}

int Date::DaysInMonth(int month, int year) const {
  if (month == 2)
    return LeapYear(year) ? 29 : 28;
  else if (month == 4 || month == 6 || month == 9 || month == 11)
    return 30;
  else
    return 31;
}

void Date::Day(int day) {
  if (day >= 1 && day <= DaysInMonth(Month(), Year())) day_ = day;
}

void Date::Month(int month) {
  if (month >= 1 && month <= 12) month_ = month;
}

void Date::Year(int year) {
  year_ = year;
}


std::string Date::String() const {
  std::vector<std::string> months{"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};
  return months[Month()-1] + " " + std::to_string(Day()) + ", " + std::to_string(Year());
}

// Test
int main() {
  Date date(29, 8, 1981);
  assert(date.String() == "August 29, 1981");
}
```

Compile & Run | Explain

Loading terminal (id_mnjjujf), please wait...