Ficha 9

Programação (L.EIC009)

Objectivos

- Exercícios versando o uso de excepções em C++.

Recursos

- Slides das aulas teóricas: HTML PDF
- Documentação online
  - `try` - `catch`
  - `throw`
  - `<stdexcept>`

1

Considere uma classe `date` para representar datas com o seguinte esqueleto:

```
class date {
private:
  int day, month, year;
public:
  date(int d, int m, int y) { ... }
  int get_day() const { return day; }
  int get_month() const { return month; }
  int get_year() const { return year; }
};
```

1.1

Defina o construtor para `date` de tal forma que no caso de ser fornecida valores inválidos para o dia e mês (`d` e `y`) seja lançada uma excepção `std::logic_error` .

Pode usar a seguinte função auxiliar (dada em aulas teóricas anteriormente) para determinar o número de dias que há num mês de determinado ano.

```
int days_in_month(int m, int y) {
  int r;
  switch (m) {
    case 2:
      // February, test for leap year
      if ( y % 4 == 0 && ( y % 100 != 0 || y % 400 == 0)) {
        r = 29;
      } else {
        r = 28;
      }
      break;
    case 1: case 3: case 5:
    case 7: case 8: case 10:
    case 12:
      r = 31; // Months with 31 days
      break;
    default:
      r = 30; // All others have 30 days
      break;
  }
  return r;
}
```

1.2

Escreva um programa que leia valores para datas e que no caso de datas inválidas que levem a uma excepção imprima uma mensagem de erro num bloco `catch` apropriadamente definido.

1.3

Substitua o uso de `std::logic_error` por uma classe `invalid_date` por si, tal que `invalid_date` seja subclasse de `std::logic_error`.

1.4

Acrescente a `date` funções membro `set_day(int d)`, `set_month(int m)` e `set_year(int y)` que permitam modificar respectivamente o dia, mês e ano de uma data.

As funções devem lançar a excepção `invalid_date` e não permitir as modificações em causa caso os argumentos sejam inválidos.

2

Em cada um dos seguintes casos, indique:

- qual o valor de `x` no final, tendo em atenção que a função membro `at` de `std::vector` lança uma excepção `std::out_of_range` no caso de receber como argumento uma posição `pos` inválida (i.e. `pos >= v.size()` para um vector `v`).
- que instruções **não** são executadas no corpo de blocos `try` ou `catch` ?

2.1

```
int x = 0;
try {
  std::vector<int> v { 5, 4, 3 };
  for (int i = 0; i < 4; i++) {
     x += v.at(i);
  }
  x *= 2;
}
catch(std::out_of_range& e) {
  x = -1;
}
```

2.2

```
int x = 0;
try {
  std::vector<int> v { 5, 4, 3 };
  for (int i = 0; i < 4; i++) {
     x += v.at(i);
  }
  x *= 2;
}
catch(std::logic_error& e) {
  x++;
}
catch(...) {
  x = -1;
}
```

2.3

```
int x = 0;
try {
  std::vector<int> v { 5, 4, 3 };
  for (int i = 0; i < 4; i++) {
     x += v.at(i);
  }
  x *= 2;
}
catch(std::runtime_error& e) {
  x ++;
}
catch(...) {
  x = -1;
}
```

2.4

```
int x = 0;
try {
  std::vector<int> v { 5, 4, 3 };
  for (int i = 0; i < 3; i++) {
     x += v.at(i);
  }
  try {
    for (int i = 3; i >= 0; i--) {
       x -= v.at(i);
    }
  }
  catch(std::out_of_range& e) {
     x += v.at(0);
  }
  x += v.at(3);
}
catch(std::runtime_error& e) {
  x --;
}
catch(...) {
  x++;
}
```