```
String getInitials(const String& name,
                   const String& surname,
                   const String& patronymic) {
     String result = String(toupper(surname[0])) + ". " +
          toupper(name[0]) + ".";
     if (patronymic.getLength() > 0) {
          result += " " + toupper(patronymic[0]) + ".";
     }
     return result;
}
```

Листинг 1 — Функция генерации ФИО

```
String getInitials(const String& name,
                   const String& surname,
                   const String& patronymic) {
     do {
          if (surname.getLength() == 0) break;
          String result = String(toupper(surname[0])) + ".";
          if (name.getLength() == 0) break;
          result += String(toupper(name[0])) + ".";
          if (patronymic.getLength() > 0) {
               result += String(" ") + toupper(patronymic[0]) + ".";
          }
          return result;
     } while (false);
     return surname + " " + name + " " + patronymic;
}
```

Листинг 2 — Функция генерации ФИО с обработкой ошибок по классической схеме

```
class StringException {
     String errorMessage;
public:
     StringException(const String& theErrorMessage)
          : errorMessage(theErrorMessage) { }
     const String& getErrorMessage() const {
          return errorMessage;
     }
};
```

Листинг 3 — Класс-исключение

```
const char String::operator[](size_t index) {
     if (index >= length) {
          throw StringException(
               String("Индекс %u больше длины %u.").format(
                    index, length)
               );
     }
     return data[index];
}
```

Листинг 4 — Модификация оператора [] в классе `String`

```
String getInitials(const String& name,
                   const String& surname,
                   const String& patronymic) {
     String result;
     try {
          result = String(toupper(surname[0])) + ". " +
               toupper(name[0]) + ".";
          if (patronymic.getLength() > 0)
               result += " " + toupper(patronymic[0]) + ".";
     } catch (const StringException& exception) {
          result = surname + " " + name + " " + patronymic;
          writeErrorToLog("Не удалось сократить имя " + result +
               ": ошибка при работе со строками: " +
               exception.getErrorMessage());
     }
     return result;
}
```

Листинг 5 — Функция генерации ФИО с обработкой ошибок с помощью `try` и `catch`

```
void printInitialList(const Person people[], size_t count) {
     for (size_t i = 0; i < count; i++) {
          const Person& person = people[i];
          try {
               const String& initials = getInitials(
                    person.getName(),
                    person.getSurname(),
                    person.getPatronymic());
               print(initials);
          } catch (const StringException& exception) {
               print(String("Ошибка обработки данных ") +
                    + person.getID() + ": " +
                    exception.getErrorMessage());
          }
     }
}
```

Листинг 6 — Функция печати списка людей с обработкой исключений

```
catch (...) {
     throw;     // Выбрасывает обрабатываемое исключение.
}
```

Листинг 7 — Блок `catch` с перехватом всех типов исключений

```
class Person {
     String name, surname, patronymic;
public:
     Person(const String& myName,
            const String& mySurname,
            const String& myPatronymic) :
     name(myName), surname(mySurname), patronymic(myPatronymic)
     try {
          // Тело конструктора.
     } catch (const StringException& exception) {
          throw PersonException(
               "Не удалось заполнить строковые поля!");
     }
};
```

Листинг 8 — Блок `try` уровня функции

```
class String {
public:
     const char operator[](const size_t index) const
          throw(StringException);
     const size_t getLength() const noexcept;
     const char* getData() const throw();
     …
};
```

Листинг 9 — Спецификация методов по возможности генерации исключений

```
namespace std {
     class exception {
     public:
          exception() noexcept;
          exception(const exception&) noexcept;
          exception& operator=(const exception&) noexcept;
          virtual ~exception();
          virtual const char* what() const noexcept;
     };
}
```

Листинг 10 — Определение класса `std::exception`