```
#include <cstring>
class String {
public:
     char *data;
     size_t length;
     void setData(const char *data);
};
void String::setData(const char *data) {
     length = strlen(data);
     this->data = new char[length + 1];
     strcpy(this->data, data);
}
String string;
string.setData("Некоторая строка");
string.setData("Тоже строка, но другая");
delete [] string.data;
```

Листинг 1 — Пример инкапсуляции полей и методов

```
#include <cstring>
class String {
private:
     char *data;
     size_t length;
public:
     void reset() {
          data = 0;
          length = 0;
     }
     void setData(const char *data) { /* Реализация та же. */ }
     void clear() {
          if (data)
               delete [] data;
          reset();
     }
     const char *getData() { return data; }
     unsigned int getLength() { return length; }
};
String string;
string.reset();
string.setData("Какая-то строка");
string.clear();
```

Листинг 2 — Пример ограничения доступа к членам объекта

```
class String {
private:
     void reset() {
          data = 0;
          length = 0;
     }
public:
     String() {
          reset();
     }
     String(const char *initial) {
          reset();
          setData(initial);
     }
     void setData(const char *data) {
          clear();
          // Здесь — старая реализация.
     }
};
String *first = new String();
String second("Начальное значение");
first->setData("Другое значение");
first->clear();
second.setData("Очередное значение");
second.clear();
delete first;
```

Листинг 3 — Пример реализации и использования конструктора

```
class String {
public:
     ~String() {
          clear();
     }
     …
};
String *first = new String();
String second("Начальное значение");
first->setData("Другое значение");
second.setData("Очередное значение");
delete first;
```

Листинг 4 — Пример реализации и использования деструктора

```
class EquatableString: public String {
public:
     EquatableString(const char *data): String(data) { }
     const bool equals(const String& other) {
          if (this == &other) {
```

```
                    return true;
               } else if (this->getLength() != other.getLength()) {
                    return false;
               }
               return !strcmp(this->getData(), other.getData());
          }
     };
     EquatableString first("12345");
     EquatableString second("54321");
     first.equals(second); // Результат: false.
     first.setData(second.getData());
     first.equals(second); // Результат: true.
```

Листинг 5 — Пример наследования

```
class DebugString : public String {
public:
     void setData(const String& string) {
          setData(string.getData());
     }
     void setData(const char *data) {
          pritnf("[%p] data changed to [%s]", this, data);
          String::setData(data);
     }
};
DebugString source("Источник"), destination("Приемник");
destination.setData(source);
```

Листинг 6 — Пример перегрузки и переопределения методов базового класса (раннее связывание)

```
void fallbackToDefaultValue(
     String& destination, String& defaultValue) {
     if (!destination.getLength()) {
          destination.setData(defaultValue.getData());
     }
}
DebugString address;
String localhost("127.0.0.1");
fallbackToDefaultValue(address, localhost);
```

Листинг 7 — Пример преобразования к базовому классу

```
class String {
public:
     virtual void setData(const char *data) {
          // Реализация прежняя.
     }
};
class DebugString : public String {
public:
     virtual void setData(const char *data) {
          // Реализация прежняя.
     }
};
```

Листинг 8 — Пример реализации виртуальных функций (позднее связывание)

```
void writeConfigValue(Printable& name, Printable& value) {
     name.print();
     printf(" = ");
     value.print();
     putchar('\n');
}
```

Листинг 9 — Пример функции печати пар: имя - значение

```
class Printable {
public:
     virtual void print() = 0;       // Реалиации нет нигде.
     virtual ~Printable() = 0;
};
Printable::~Printable() { }     //  Должно быть не inline.
```

Листинг 10 — Пример определения чисто виртуальных функций

```
class String : public Printable {
public:
     virtual void print() {
          if (data)
               printf("%s", data);
     }
};
class Date : public Printable {
public:
     virtual void print() {
          pritnf("%u.%u.%u", day, month, year);
     }
};
writeConfigValue(String("end_of_days"), Date(21, 12, 2012));
```

Листинг 11 — Пример наследников с реализацией виртуальных функций