```
String(const String& copiedOne) {
     setData(copiedOne.data);
}
...
void function(String string);
String first("Имя мое Легион, потому что нас много.");
String second(first);
function(first);
```

Листинг 1 — Пример использования конструктора копирования

```
String(String&& movedOne) {
     this->data = movedOne.data;
     this->length = movedOne.length;
     movedOne.data = 0;
     movedOne.length = 0;
}
…
String function() {
     return String("We have to move mountains, "
                   "because our enemies move planets!");
}
String result = function()
```

Листинг 2 — Пример использования конструктора перемещения

```
class String {
     …
public:
     char operator[](size_t index) const { return data[index];
}
…
String game("Mass Effect");
putchar(game[0]);
putchar(game.operator[](0));
```

Листинг 3 — Перегрузка оператора []

```
int operator+(int left, int right);
...
const String operator+(const String& left, const String& right)
{
     String result(left);
     result.concatenateWith(right.data, right.length);
     return result;
}
...

class String {
```

```
        …
        friend
        const String operator+(const String& left, const String&
right);
}
…
String name("Шепард");
String fullName = "Капитан " + name;
```

Листинг 4 — Перегрузка оператора +

```
friend bool operator==(const String& object, const char *ntcs);
friend bool operator==(const char *ntcs, const String& object);
```

Листинг 5 — Перегрузка оператора ==

```
operator bool() const {
        return data && length;
}
```

Листинг 6 – Перегрузка оператора приведения типов

```
String& String::operator=(const String& rightSide) {
        setData(rightSide.data);
        return *this;
}
String& String::operator=(String&& rightSide) {
        this.data = rightSide.data;
        this.length = rightSide.length;
        rightSide.data = 0;
        rightSide.length = 0;
        return *this;
}
```

Листинг 7 — Перегрузка оператора присвоения

```
class Functor {
public:
        void operator() (double, int) { }
};
…
Functor f;
f(3.14, 0);
```

Листинг 8 — Пример функтора

```
class Object {
public:
        static void* operator new (size_t) {
                Object *created = ::new Object;
                if (tail) {
                        tail->next = created;
```

```
7                } else {
8                     list = tail = created;
9                }
10               created->next = 0;
11               return created;
12          }
13          static void operator delete (void* object) nothrow {
14          }
15          static void cleanUp() {
16               Object *current = list;
17               while (current) {
18                    Object *victim = current;
19                    current = current->next;
20                    ::delete victim;
21               }
22               list = tail = 0;
23          }
24     private:
25          Object* next;
26     private:
27          static Object *list;
28          static Object *tail;
29     };
30     Object* Object::list = 0;
31     Object* Object::tail = 0;
32     Object *first = new Object;
33     Object *second = new Object;
34     Object::cleanUp();
```

Листинг 9 — Перегрузка операторов new и delete

```
1      class IClickDelegate {
2      public:
3           virtual void invoke() = 0;
4           virtual ~IClickDelegate() = 0;
5      };
6      IClickDelegate::~IClickDelegate() { }
7      class Button {
8      public:
9           Button() :clickEvent(0) { }
10          virtual ~Button() {
11               if (clickEvent) delete clickEvent;
12          }
13          void click() {
14               if (clickEvent) clickEvent->invoke();
15          }
16          void setClickHandler(IClickDelegate *eventHandler) {
17               clickEvent = eventHandler;
```

```
18          }
19      private:
20          IClickDelegate *clickEvent;
21      };
22      class Window;
23      class WindowClickDelegate : public IClickDelegate {
24      public:
25          typedef void (Window::*ButtonClickEventHandler)(void);
26          WindowClickDelegate(
27              Window& window, ButtonClickEventHandler handler)
28              : window(window), handler(handler) { }
29          virtual void invoke() {
30              (window.*handler)();
31          }
32      private:
33          Window& window;
34          ButtonClickEventHandler handler;
35      };
36      class Window {
37      public:
38          Window() : okButton(), cancelButton() {
39              okButton.setClickEvent(new WindowClickDelegate(
40                  *this, &Window::onOKButtonClicked));
41              cancelButton.setClickEvent(new WindowClickDelegate(
42                  *this, &Window::onCancelButtonClicked));
43          }
44          void onOKButtonClicked() {
45              // Обработать нажатие на кнопку «OK».
46          }
47          void onCancelButtonClicked() {
48              // Обработать нажатие на кнопку «Отмена».
49          }
50      public:
51          Button okButton, cancelButton;
52      };
53      …
54      Window window;
55      window.okButton.click();
```

Листинг 10 — Пример использования указателя на члены класса