ЛЕКЦИЯ 04
КОНТЕЙНЕРЫ. STRING

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

ЛЕКТОР ФУРМАВНИН С.А.

ИЗУЧИТЕ КОД

```
int main()
{
    vector<Date> e;
    copy(istream_iterator<Date>(cin), istream_iterator<Date>(),
            back_inserter(e));
    vector<Date>::iterator   first   =   find(e.begin(),   e.end(),
"01/01/2015");
    vector<Date>::iterator   last   =   find(e.begin(),   e.end(),
"12/31/2015");
    *last = "12/30/2015";
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
    e.insert(--e.end(), TodaysDate());
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
}
```

Сколько ошибок вы можете в нем найти за 10 минут? Выпишите их на лист бумаги.

АНАЛИЗ ОШИБОК

```
int main()
{
    vector<Date> e;
    copy(istream_iterator<Date>(cin), istream_iterator<Date>(),
            back_inserter(e));
    vector<Date>::iterator   first   =   find(e.begin(),   e.end(),
"01/01/2015");
    vector<Date>::iterator   last   =   find(e.begin(),   e.end(),
"12/31/2015");
    *last = "12/30/2015";
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
    e.insert(--e.end(), TodaysDate());
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
}
```

Сколько ошибок вы можете в нем найти за 10 минут? Выпишите их на лист бумаги.

АНАЛИЗ ОШИБОК

```
vector<Date>::iterator first = find(e.begin(), e.end(),
"01/01/2015");
    vector<Date>::iterator last = find(e.begin(), e.end(),
"12/31/2015");
    *last = "12/30/2015";
```

Ошибка: Это присваивание может быть некорректным, поскольку `last` может оказаться равным `e.end()`.

АНАЛИЗ ОШИБОК

```
int main()
{
    vector<Date> e;
    copy(istream_iterator<Date>(cin), istream_iterator<Date>(),
            back_inserter(e));
    vector<Date>::iterator    first    =    find(e.begin(),    e.end(),
"01/01/2015");
    vector<Date>::iterator    last    =    find(e.begin(),    e.end(),
"12/31/2015");
    *last = "12/30/2015";
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
    e.insert(--e.end(), TodaysDate());
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
}
```

Сколько ошибок вы можете в нем найти за 10 минут? Выпишите их на лист бумаги.

АНАЛИЗ ОШИБОК

```
copy(first, last, ostream_iterator<Date>(cout,"\n"));
```

Ошибка: `[first, last)` может не быть корректным диапазоном. На самом деле `first` может находиться после `last`.

АНАЛИЗ ОШИБОК

```
int main()
{
    vector<Date> e;
    copy(istream_iterator<Date>(cin), istream_iterator<Date>(),
            back_inserter(e));
    vector<Date>::iterator   first   =   find(e.begin(),   e.end(),
"01/01/2015");
    vector<Date>::iterator   last   =   find(e.begin(),   e.end(),
"12/31/2015");
    *last = "12/30/2015";
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
    e.insert(--e.end(), TodaysDate());
    copy(first, last, ostream_iterator<Date>(cout,"\n"));
}
```

Сколько ошибок вы можете в нем найти за 10 минут? Выпишите их на лист бумаги.

АНАЛИЗ ОШИБОК

```
e.insert(--e.end(), TodaysDate());
```

Первая ошибка: выражение `--e.end()` вполне может оказаться неверным. Причина довольна проста: в популярных реализациях STL `vector<Date>::iterator` зачастую представляет собой `Date*`, а язык C++ не позволяет изменять временные переменные встроенного типа.

```
Date* f(); // Функция, возвращающая Date*
p = --f(); // Ошибка
P = f() - 1; // OK
```

АНАЛИЗ ОШИБОК

```
e.insert(--e.end(), TodaysDate());
```

Вторая ошибка: Если контейнер `e` пуст, то любые попытки получить итератор, указывающий на позицию перед `e.end()`, будут некорректны.

РЕЗЮМИРУЕМ

При использовании итераторов всегда задавайте себе эти вопросы:

1. Корректность значений. Возможно ли разыменование данного итератора?
2. Корректность времени жизни объекта. Остался ли корректен используемый вами итератор после выполнения предыдущих действий?
3. Корректность диапазона. Представляет ли пара итераторов корректный диапазон?
4. Некорректные встроенные операции. Не пытается ли код модифицировать временный объект встроенного типа?

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

1. Что означает «нечувствительный к регистру»?

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

1. Что означает «нечувствительный к регистру»?

Что это означает, в действительности полностью зависит от приложения и языка. Например, многие языки вообще не поддерживают различные регистры символов. В ходе ответа на этот вопрос вам надо также определить, считаете ли вы акцентированные символы эквивалентными неакцентированным (например, одинаковы ли символы в строке «aáâãä»), и рассмотреть многие другие вопросы. Ответ на данный вопрос представляет собой руководство, каким образом следует реализовать нечувствительность к регистру стандартных строк в любом смысле, который вложите в это понятие.

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

2. Напишите класс `ci_string`, идентичный стандартному `std::string`, но который является нечувствительным к регистру символов в том же смысле, что и распространенное расширение библиотеки C++ `stricmp()`. То есть мы хотим получить это:

```
ci_string s("AbCdE");
// Нечувствительно к регистру
assert( s == "abcde" );
assert( s == "ABCDE" );
// Остается чувствительно к регистру
assert(strcmp(s.c_str(),"AbCdE") == 0);
assert(strcmp(s.c_str(),"abcde") != 0);
```

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

3. Разумно ли делать нечувствительность к регистру свойством объекта?

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

3. Разумно ли делать нечувствительность к регистру свойством объекта?

Зачастую более полезно, когда нечувствительность к регистру является свойством функции сравнения, а не объекта. Например, рассмотрим код:

```
string a = "aaa";
ci_string b = "aAa";
if (a == b) /* ... */
```

Каким должен быть результат сравнения?

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

Насколько практичен наш класс `ci_string`?

СТРОКИ. НЕЧУВСТВИТЕЛЬНЫЕ К РЕГИСТРУ

Насколько практичен наш класс `ci_string`?

- Безопасно ли наследовать `ci_char_traits` от `char_traits<char>`?
- Почему этот код вызовет ошибку компиляции?

```
ci_string s = "abc";
std::cout << s << std::endl;
```

- Что можете сказать об использовании других операторов (+, += и т.д.) и о сочетании `string` и `ci_string` в качестве их аргументов? Например,

```
std::string a = "aaa";
ci_string b = "bbb";
std::string c = a + b;
```

ДОМАШНЕЕ ЗАДАНИЕ

Изучить код, реализующий решение второго вопроса и покритиковать его. Подумайте над ответом на 3 вопрос.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Бьерн Страуструп, Язык программирования C++/ ред. А. Боборыкин. – 4-е изд. - Москва: Издательство БИНОМ, 2023. – 1213 с.
2. Скотт Мейерс, Эффективное использование C++. 55 верных способов улучшить структуру и код ваших программ / ред. Д.А. Мовчан – 3-е изд. – Москва: ДМК Пресс, 2017. – 300 с.
3. Скотт Мейерс, Наиболее эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов / ред. Д.А. Мовчан – 3-е изд. – Москва: ДМК Пресс, 2016. – 298 с.