ЛЕКЦИЯ 03
КОНТЕЙНЕРЫ. STRING

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

ЛЕКТОР ФУРМАВНИН С.А.

СТРОКОВЫЕ ЛИТЕРАЛЫ

Самый простой код, который только возможен:

```
std::cout << “Hello, world!” << std::endl;
```

СТРОКОВЫЕ ЛИТЕРАЛЫ

Самый простой код, который только возможен:

```
std::cout << “Hello, world!” << std::endl;
```

“Hello, world!” – это строковый литерал, т.е. константа времени компиляции

СТРОКОВЫЕ ЛИТЕРАЛЫ

Самый простой код, который только возможен:

```
std::cout << “Hello, world!” << std::endl;
```

“Hello, world!” – это строковый литерал, т.е. константа времени компиляции

Ещё примеры литералов `0x001`, `076`, `1.6e-3f`, `true`, `“MFUA”`, `‘s’`

Примеры строковых литералов: `u8”Hello”`, `L”Hello”`, `U”Hello”`, `u”Hello”`

СТРОКОВЫЕ ЛИТЕРАЛЫ

Самый простой код, который только возможен:

```
std::cout << “Hello, world!” << std::endl;
```

“Hello, world!” – это строковый литерал, т.е. константа времени компиляции

Ещё примеры литералов `0x001`, `076`, `1.6e-3f`, `true`, `“MFUA”`, `‘s’`

Примеры строковых литералов: `u8”Hello”`, `L”Hello”`, `U”Hello”`, `u”Hello”`

ПРЕДСТАВЛЕНИЕ СТРОК

Строки в C++ исторически завершаются нулевым символом

H	e	l	l	o	,		w	o	r	l	d	!	\0

Строка могла быть устроена иначе. Например предваряться размером.

\13	H	e	l	l	o	,		w	o	r	l	d	!

РАБОТА СО СТРОКАМИ

```
const char* cinv = “Hello, world”;
char cmut[] = “Hello, world”;
char* cheap = new(какой-то размер);
strcpy(cheap, cinv);
cheap = cinv;
cinv = 0;
cmut = cheap;
```

РАБОТА СО СТРОКАМИ

```
const char* cinv = “Hello, world”;
char cmut[] = “Hello, world”; // копирование
char* cheap = new(какой-то размер);
strcpy(cheap, cinv); // копирование
cheap = cinv; // ошибка и потенциал утечки
cinv = 0;
cmut = cheap; // ошибка компиляции
```

ПРОБЛЕМАТИКА

- Реаллокация памяти при добавлении и удалении символа
- Реаллокация памяти при токенизации строки
- Возможная утеря конца строки = проблемы с безопасностью

РЕШЕНИЕ ПРОБЛЕМ

Объектно-ориентированный подход.

РЕШЕНИЕ ПРОБЛЕМ

Объектно-ориентированный подход.

Написать класс, который:

- ~~Реаллокация памяти при добавлении и удалении символа~~
- Реаллокация памяти при токенизации строки
- Выделяет «запас» по памяти

РЕШЕНИЕ ПРОБЛЕМ

Объектно-ориентированный подход.

Написать класс, который:

- ~~Реаллокация памяти при добавлении и удалении символа~~
- Реаллокация памяти при токенизации строки
- Выделяет «запас» по памяти
- ~~Возможная утеря конца строки = проблемы с безопасностью~~
- Хранит настоящий размер строки и инкапсулирует его

ПРОЕКТ СТРОКИ

ПРОЕКТ СТРОКИ

```
size_t size;
char* str;
```

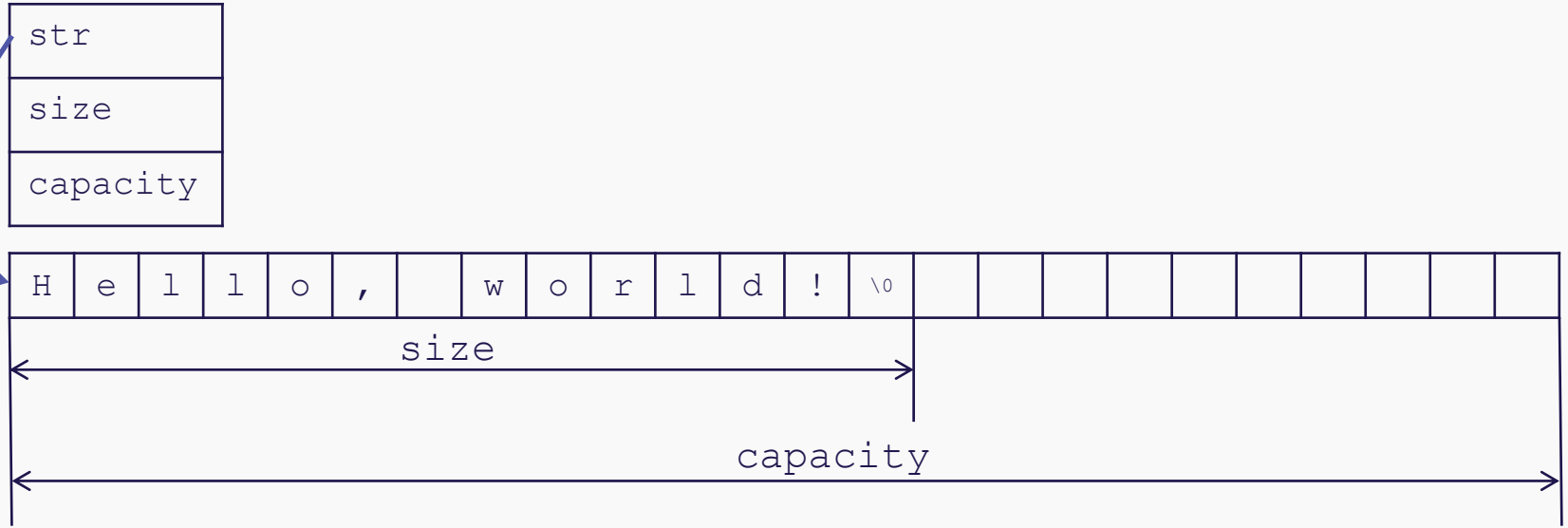
ПРОЕКТ СТРОКИ

```
size_t size;
char* str;
```

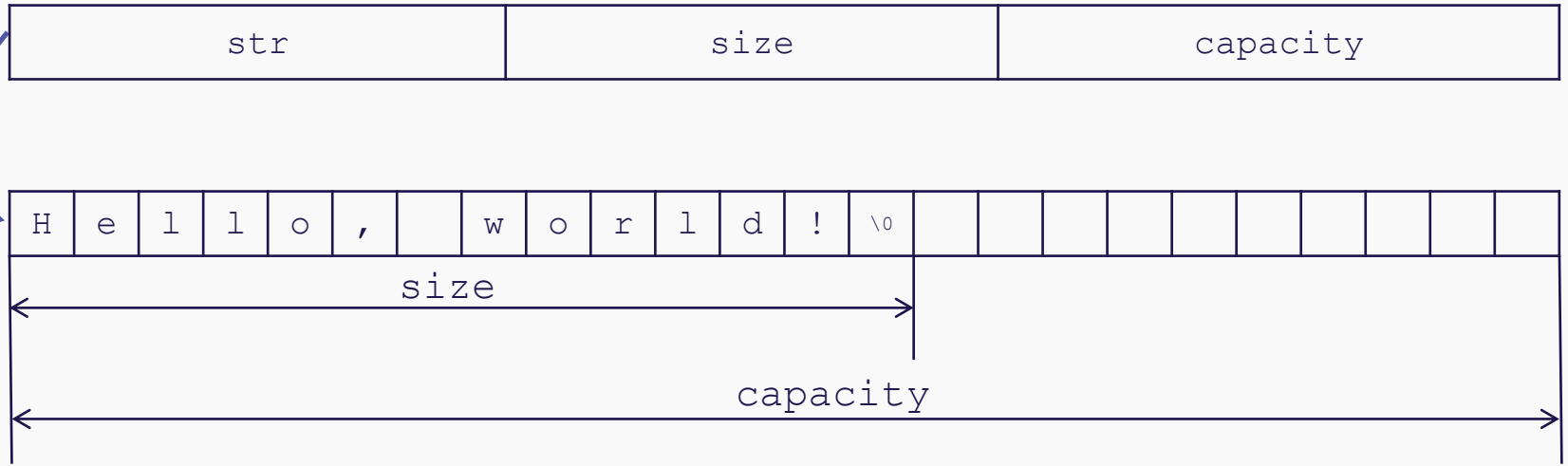
ПРОЕКТ СТРОКИ

```
size_t size;
char* str;
size_t capacity;
```

ПРОЕКТ СТРОКИ

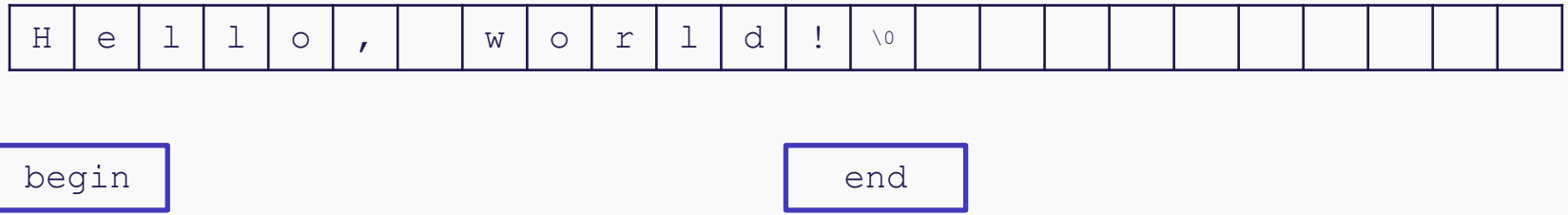


ПРОЕКТ СТРОКИ



ПРОЕКТ СТРОКИ

```
size_t size;
char* str;
size_t capacity;
```



ИТЕРАТОРЫ В КОНТЕЙНЕРАХ И ПОСЛЕДОВАТЕЛЬНОСТЯХ

`begin` – это итератор, который указывает на первый элемент контейнера или последовательности. Инкремент этого итератора вернет итератор на следующий элемент, а декремент – UB.

ИТЕРАТОРЫ В КОНТЕЙНЕРАХ И ПОСЛЕДОВАТЕЛЬНОСТЯХ

`begin` – это итератор, который указывает на первый элемент контейнера или последовательности. Инкремент этого итератора вернет итератор на следующий элемент, а декремент – UB.

`end` – это итератор на конец контейнера или последовательности. Декремент этого итератора вернет итератор на последний элемент контейнера или последовательности, а инкремент – UB.

ИТЕРАТОРЫ В КОНТЕЙНЕРАХ И ПОСЛЕДОВАТЕЛЬНОСТЯХ

`rbegin` – это итератор, который указывает на последний элемент контейнера или последовательности. Инкремент этого итератора вернет итератор на предыдущий элемент, а декремент – UB.

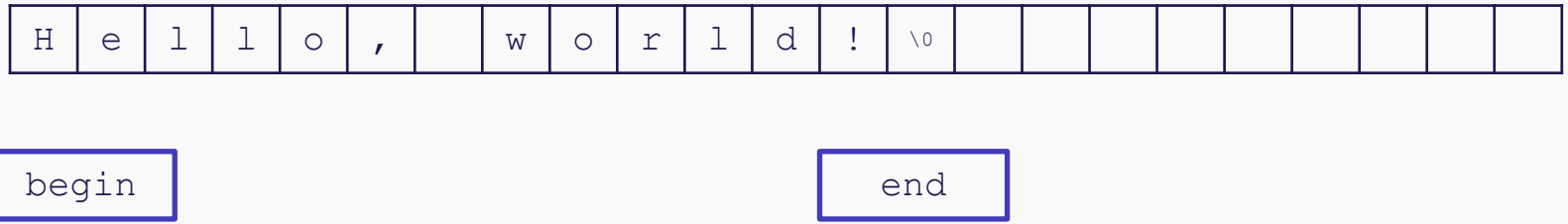
ИТЕРАТОРЫ В КОНТЕЙНЕРАХ И ПОСЛЕДОВАТЕЛЬНОСТЯХ

`rbegin` – это итератор, который указывает на последний элемент контейнера или последовательности. Инкремент этого итератора вернет итератор на предыдущий элемент, а декремент – UB.

`rend` – это итератор на начало контейнера или последовательности. Декремент этого итератора вернет итератор на первый элемент контейнера или последовательности, а инкремент – UB.

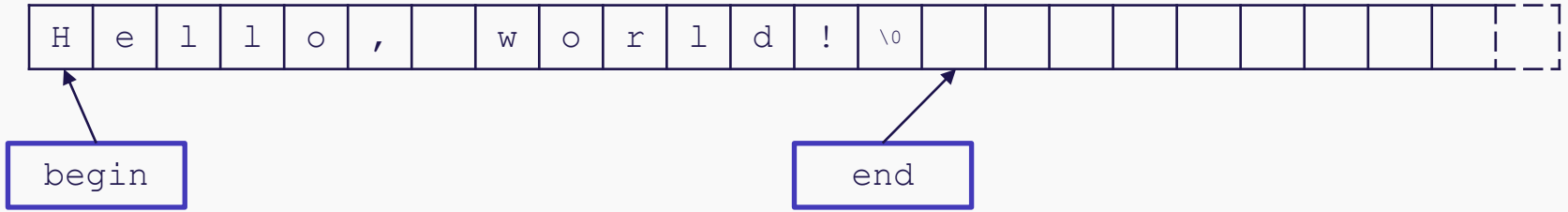
ПРОЕКТ СТРОКИ

```
size_t size;
char* str;
size_t capacity;
```



ПРОЕКТ СТРОКИ

```
size_t size;
char* str;
size_t capacity;
```



ОСНОВНЫЕ МЕТОДЫ СТРОКИ

Какие конструкторы должны быть реализованы?

ОСНОВНЫЕ МЕТОДЫ СТРОКИ

Какие конструкторы должны быть реализованы?

- Конструктор по умолчанию
- Преобразующий конструктор от const char*
- Конструктор копирования
- Конструктор перемещения

ОСНОВНЫЕ МЕТОДЫ СТРОКИ

Какие конструкторы должны быть реализованы?

- Конструктор по умолчанию
- Преобразующий конструктор от const char*
- Конструктор копирования
- Конструктор перемещения

Наличие нетривиального конструктора подразумевает необходимость написать

ОСНОВНЫЕ МЕТОДЫ СТРОКИ

Какие конструкторы должны быть реализованы?

- Конструктор по умолчанию
- Преобразующий конструктор от const char*
- Конструктор копирования
- Конструктор перемещения

Наличие нетривиального конструктора подразумевает необходимость написать **деструктор**

ОСНОВНЫЕ МЕТОДЫ СТРОКИ

Какие конструкторы должны быть реализованы?

- Конструктор по умолчанию
- Преобразующий конструктор от const char*
- Конструктор копирования
- Конструктор перемещения

Наличие нетривиального конструктора подразумевает необходимость написать **деструктор**

ДОМАШНЕЕ ЗАДАНИЕ

Написать свой класс string. Помимо оговоренного на лекции определить следующие методы:

1. `empty` – возвращает `false` для пустой строки и `true` для непустой.
2. `size` – возвращает длину строки
3. `capacity` – возвращает количество символов, которые могут быть записаны в выделенную уже память
4. `reserve` – гарантирует запись заданного количества символов без реаллокации
5. `shrink_to_fit` – уменьшает размер выделенной памяти до фактического размера строки

Это минимум на оценку УДОВЛЕТВОРИТЕЛЬНО

ДОМАШНЕЕ ЗАДАНИЕ

Написать свой класс string. Помимо оговоренного на лекции определить следующие методы:

1. `push_back` – добавляет символ в конец строки
2. `erase` – удаляет символ по итератору или между двумя итераторами
3. `insert` – вставляет символ в указанную позицию в заданном количестве символ или по итератору
4. `append` – добавляет набор символов в конец строки
5. `substr` – возвращает подстроку начиная с номера указанного символа в указанном количестве

Это минимум на оценку ХОРОШО

ДОМАШНЕЕ ЗАДАНИЕ

Написать свой класс string. Помимо оговоренного на лекции определить следующие методы:

1. `find` – находит первое вхождение заданной подстроки и возвращает номер её первого элемента
2. `rfind` – находит последнее вхождение заданной подстроки и возвращает номер её первого элемента
3. `replace` – заменяет в заданном диапазоне символы новыми символами

Это минимум на оценку ОТЛИЧНО

ВРЕМЯ ДЛЯ ВАШИХ ВОПРОСОВ

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.