ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Департамент программной инженерии

Алгоритмы и структуры данных

# Семинар №4. 2021-2022 учебный год

**Нестеров Роман Александрович**, *ДПИ ФКН и НУЛ ПОИС*

**Бессмертный Александр Игоревич**, *ДПИ ФКН*

Higher School of Economics

Faculty of Computer science

# Veni, vidi, vici

# Veni, vidi, vici

# PCF, WA, WA, …, OK!!!

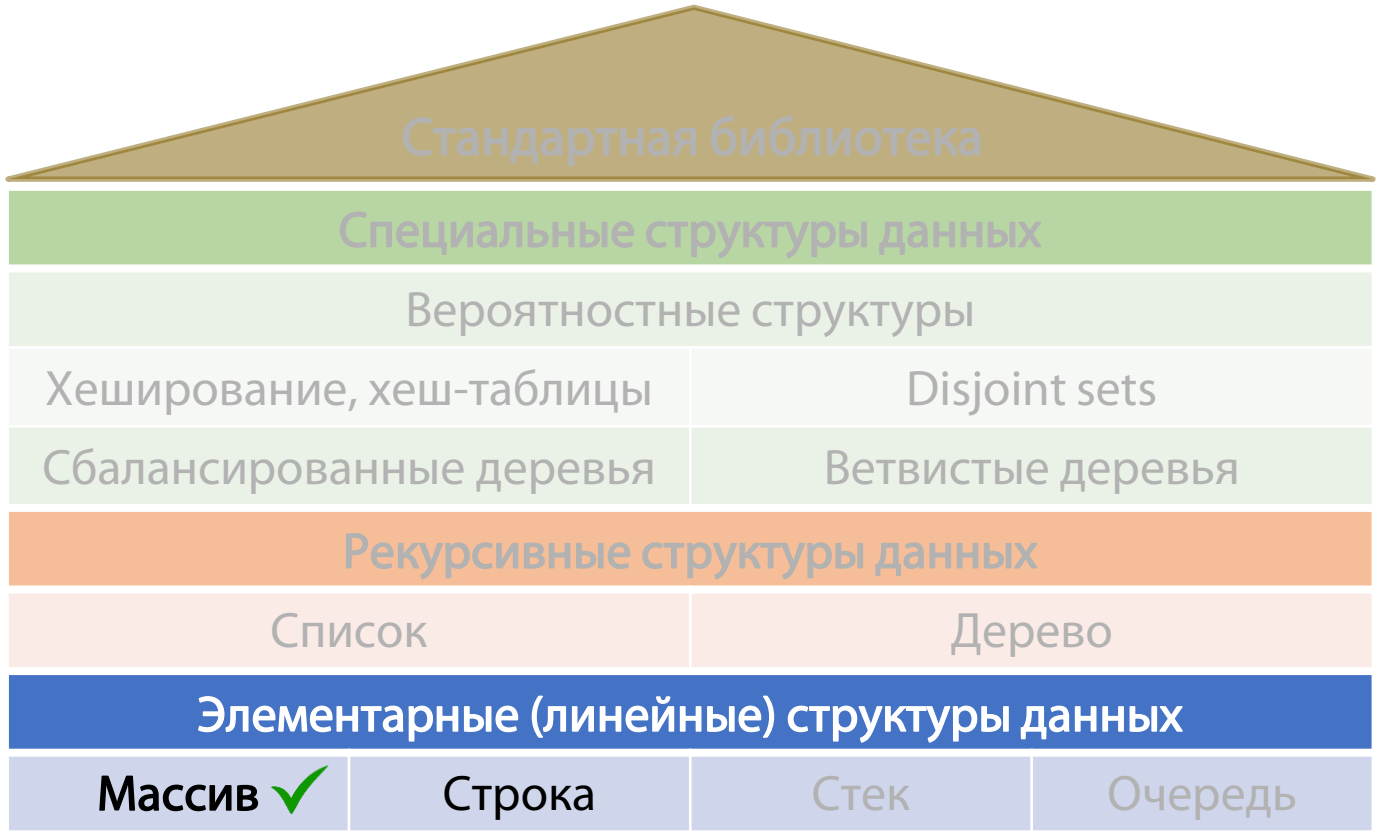
# План

- Строки `C-style` – массивы символов
- Строки как объекты класса `std::string`
- Представление строк `std::string_view` (C++ 17)
- Потоковые классы `std::istringstream`, `std::ostringstream`

# Где мы?


Стандартная библиотека
Специальные структуры данных
Вероятностные структуры
Хеширование, хеш-таблицы
Disjoint sets
Сбалансированные деревья
Ветвистые деревья
Рекурсивные структуры данных
Список
Дерево
Элементарные (линейные) структуры данных
Массив
Строка
Стек
Очередь

# Где мы?


Стандартная библиотека
Специальные структуры данных
Вероятностные структуры
Хеширование, хеш-таблицы
Disjoint sets
Сбалансированные деревья
Ветвистые деревья
Рекурсивные структуры данных
Список
Дерево
Элементарные (линейные) структуры данных
Массив
Строка
Стек
Очередь

# Строки C-style

# Одномерный массив символов

```
#include <iostream>

int main() {
    char string[] = "testString";
    std::cout << sizeof(string) << ' ';

    for (size_t i = 0; i < sizeof(string); ++i) {
        std::cout << static_cast<int>(string[i]) << ' ';
    }

    return 0;
}
```

```
11 116 101 115 116 83 116 114 105 110 103 0
```

# Нуль-терминатор \0

```
#include <iostream>

int main() {
    char string[100] = "Short string";
    string[13] = 'p';
    std::cout << string << '\n';

    char string1[] = "Short string";
    string1[12] = 'p';
    std::cout << string1;

    return 0;
}
```

```
Short string
Short stringp╠╠╠╠╠╠╠╠╠╠Short string
```

# Нуль-терминатор \0

```
#include <iostream>

int main() {
    char string[100] = "Short string";
    string[13] = 'p';
    std::cout << string << '\n';

    char string1[] = "Short string";
    string1[12] = 'p';
    std::cout << string1;

    return 0;
}
```

```
Short string
Short stringp╠╠╠╠╠╠╠╠╠╠Short string
```

# Undefined или unspecified behavior

# Undefined или unspecified behavior

```
int i = 5;
i = ++i + ++i;
```

i = 12?
i = 13?
i = 14?

# Undefined или unspecified behavior

```
int i = 5;
i = ++i + ++i;
```

i = 12?
i = 13?
i = 14?

```
int f1() { std::cout << "f1 "; return 1; }
int f2() { std::cout << "f2 "; return 2; }

int main() {
    int i = 0;
    i = f1() + f2();

    return 0;
}
```

# Undefined или unspecified behavior

```
int i = 5;
i = ++i + ++i;
```

i = 12?
i = 13?
i = 14?

```
int f1() { std::cout << "f1 "; return 1; }
int f2() { std::cout << "f2 "; return 2; }

int main() {
    int i = 0;
    i = f1() + f2();

    return 0;
}
```

f1 f2 или f2 f1?

# Функции для работы со строками C-style

```
#include <iostream>
#include <cstring>

int main() {
    char string1[] = "Short string";
    char string2[5];

    strcpy_s(string2, string1);

    std::cout << string2;

    return 0;
}
```

???

# Функции для работы со строками C-style

```
#include <iostream>
#include <cstring>

int main() {
    char string1[] = "Short string";
    char string2[5];

    strcpy_s(string2, string1);
    std::cout << string2;

    return 0;
}
```

```
...
```

```
#include <iostream>
#include <cstring>

int main() {
    char string[225] = "Short string";

    std::cout << sizeof(string) << ' ';
    std::cout << strlen(string);

    return 0;
}
```

```
225 12
```

# Функции для работы со строками C-style

- Конкатенация строк `strcat, strncat`
- Сравнение строк с учетом регистра/без учета регистра `strcmp, strcmpi`
- Сравнение строк до заданного количества символов `strncmp`
- …

# Строки `std::string`

# Класс `std::string`

- Создание и манипулирование строками в форме, интуитивно понятной разработчику
- Автоматическое управление памятью
- Конструкторы, деструкторы и перегруженные операторы

```
#include <string>
```

# Создание строк

- Конструктор по умолчанию
- Конструктор копирования

- Создание строки на основе данной, начиная с некоторой позиции
- Создание строки на основе данной строки `char *`
- Создание строки на основе символа и количества его вхождений

# Создание строк

```cpp
#include <iostream>
#include <string>
using std::string;

int main() {

    string s1("Test string");
    string s2(s1, 4);
    std::cout << s2 << ' ';

    string s3("Test string #236");
    string s4(s3, 2, 10);
    std::cout << s4;

    return 0;
}
```

```
 string st string
```

```cpp
#include <iostream>
#include <string>
using std::string;

int main() {

    string s5(25, 'e');
    std::cout << s5;

    return 0;
}
```

```
eeeeeeeeeeeeeeeeeeeeeeeee
```

# Ввод строк getline

```
#include <iostream>
#include <string>
using std::string;

int main() {
    std::cout << "Enter title: ";
    std::string title;
    std::cin >> title;

    std::cout << "Enter pages: " << '\n';
    std::string pages;
    std::cin >> pages;

    std::cout << title << ' ' << pages;
    return 0;
}
```

```
Enter title: Adventures of
Captain Flint
Enter pages:
Adventures of
```

# Ввод строк getline

```
#include <iostream>
#include <string>
using std::string;

int main() {
    std::cout << "Enter title: ";
    std::string title;
    std::getline(std::cin, title);

    std::cout << "Enter pages: ";
    std::string pages;
    std::getline(std::cin, pages);

    std::cout << title << ' ' << pages;
    return 0;
}
```

```
Enter title: Adventures of Captain Flint
Enter pages: 256
Adventures of Captain Flint
256
```

# Ввод строк getline

```
#include <iostream>
#include <string>
using std::string;

int main() {
    std::cout << "Enter number: ";
    int x;
    std::cin >> x;

    std::cout << "Enter string: " << '\n' ;
    std::string s;
    getline(std::cin, s);

    std::cout << x << '\n' << s;
    return 0;
}
```

```
Enter number: 455558
Enter string:
455558
```

# Ввод строк getline

```
#include <iostream>
#include <string>
using std::string;

int main() {
    std::cout << "Enter number: ";
    int x;
    std::cin >> x;
    std::cin.ignore(65535, '\n');
    std::cout << "Enter string: ";
    std::string s;
    getline(std::cin, s);

    std::cout << x << '\n' << s;
    return 0;
}
```

```
Enter number: 555589
Enter string: test string hi
555589
test string hi
```

# Длина, максимальный размер и емкость строк

```cpp
#include <iostream>
#include <string>

int main() {
    std::string test("New string 896");

    std::cout << test.length() << '\n';
    std::cout << test.max_size();

    return 0;
}
```

```
14
2147483647
```

# Длина, максимальный размер и емкость строк

```cpp
#include <iostream>
#include <string>

int main() {
    std::string test("New string 896");

    std::cout << test.length() << '\n';
    std::cout << test.max_size();

    return 0;
}
```

```
14
2147483647
```

```cpp
#include <iostream>
#include <string>

int main() {
    std::string test("New string 896");

    std::cout << test.capacity() << '\n';

    test += "Another String 999";

    std::cout << test.capacity();
    return 0;
}
```

```
15
47
```

# Посимвольный доступ

```cpp
#include <iostream>
#include <string>

int main() {
    std::string test("New string 896");

    std::cout << test[5] << '\n';
    std::cout << test[45];

    return 0;
}
```

```
t
↑
```

# Посимвольный доступ

```cpp
#include <iostream>
#include <string>

int main() {
    std::string test("New string 896");

    std::cout << test[5] << '\n';
    std::cout << test[45];

    return 0;
}
```

```
t
↑
```

```cpp
#include <iostream>
#include <string>

int main() {
    std::string test("New string 896");

    std::cout << test.at(5) << '\n';
    std::cout << test.at(45);

    return 0;
}
```

# Варианты конкатенации

```
#include <iostream>
#include <string>

int main()
{
    std::string test("Test1");

    test += std::string(" Test2");
    test.append(" " + test).append(" Test 3");

    std::cout << test;

    return 0;
}
```

```
Test1 Test2 Test1 Test2 Test 3
```

```
#include <iostream>
#include <string>

int main()
{
    std::string test("Test1");

    test.append(6, 'g').append(3,'e');

    std::cout << test;

    return 0;
}
```

```
Test1ggggggeee
```

# Другие функции

- Сравнение строк ==, !=, >, <=, >, >=
- Выделение подстроки `substr()`

- Поиск вхождений символов/строк
  - `find_first_of, find_first_not_of`
  - `find_last_of, find_last_not_of`
- `...`

# Представление строки `string_view`

# Константные строки

```
#include <iostream>
#include <string>

int main() {
    char string1[] = "Test string";

    std::string string2(string1);
    std::string string3(string2);

    return 0;
}
```

Строка `"Test string"` копируется **4 раза**

# Представление строки `string_view`

Копия исходной строки **не создается**.

```
#include <iostream>
#include <string_view>

int main() {

    char string1[] = "Test string";
    std::string_view string2(string1);

    string1[8] = 'o';

    std::cout << string2;

    return 0;
}
```

```
Test strong
```

# Модификация представления

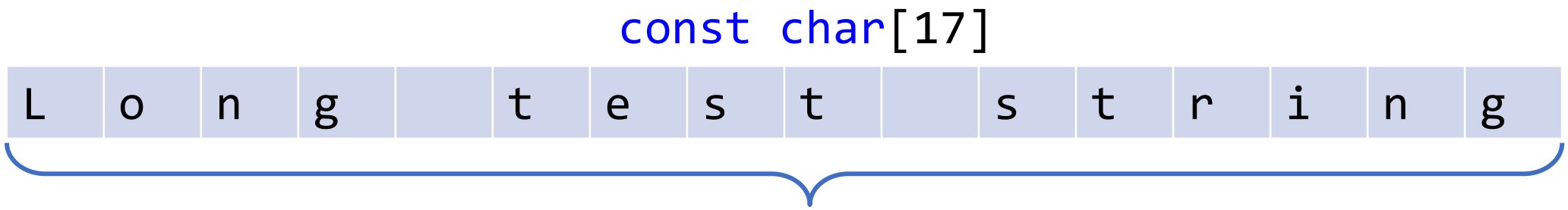
`string_view` **не владеет** исходной строкой

```cpp
#include <iostream>
#include <string_view>

int main() {

    std::string_view string2("Long Test String");

    string2.remove_prefix(5);
    std::cout << string2 << '\n';

    string2.remove_suffix(4);
    std::cout << string2;
    return 0;
}
```

```
Test String
Test St
```

# Модификация представления

`string_view` **не владеет** исходной строкой



# Модификация представления

`string_view` **не владеет** исходной строкой

```
string2.remove_suffix(11)
```


```
std::string_view string2("Long Test String")
```

# Выход исходной строки из области видимости

Неопределенные результаты при попытке вывода `string_view`

```cpp
#include <iostream>
#include <string>
#include <string_view>

std::string_view func() {
    std::string str;
    std::getline(std::cin, str);

    std::string_view sv(str);
    std::cout << sv << ' ';

    return sv;
}
```

```cpp
int main() {
    std::string_view sv;
    sv = func();

    for (size_t i = 0; i < sv.length(); ++i) {
        std::cout << sv[i] << ' ';
    }

    return 0;
}
```

```
Example string  ╠ ╠ ╠ ╠ ╠ ╠ ╠ ╠ ╠ ╠ ╠ ╠
```

# Преобразование `string_view` в `string`

Необходимо **явное** преобразование в строку

```
#include <iostream>
#include <string>
#include <string_view>

void print(std::string str) {
    std::cout << str << '\n';
}
```

```
int main() {
    char s[] = “Test string";
    std::string_view sv(s);
    std::string str1(sv);

    print(str1);
    print(static_cast<std::string>(sv));

    s[3] = 'j';
    print(str1);
    print(static_cast<std::string>(sv));
    return 0;
}
```

```
Test string Test string Test string Tesj string
```

# string_view

- Обеспечивает представление строки **без создания** своей собственной копии
- **Не владеет** (не способно изменить) исходной строкой (содержит только указатель на исходную строку и ее длину)
- Представление может быть модифицировано

# Строковые потоки

# Строковые потоки `sstream`

- Использование строк в качестве потоков ввода/вывода
- Строковые потоки используют строки в качестве буфера, содержащего последовательность символов
- Поддерживают операторы вставки (<<) и извлечения (>>)


```
#include <sstream>
```

# Подсчет количества слов в строке

```cpp
#include <iostream>
#include <sstream>
#include <string>

int countWords(std::string s)
{
    std::stringstream stream(s);
    std::string word;

    int wordsCount = 0;
    while (stream >> word)
    {
        ++wordsCount;
    }

    return wordsCount;
}
```

```cpp
int main() {

    std::string str;
    std::getline(std::cin, str);

    std::cout << countWords(str);

    return 0;
}
```

# Конверсия число<->строка

```
int main() {
    std::string s1 = "1234pl5.69";
    std::string s2 = "some string";
    std::stringstream myStream;

    myStream << s1;
    int x = 0;
    myStream >> x;
    std::cout << x << ' ';
    myStream.str("");

    myStream << s2;
    myStream >> x;
    std::cout << x;

    return 0;
}
```

```
1234 0
```

# Конверсия число<->строка

```
int main() {
    int var1 = 32767;
    double var2 = 5559.00001;
    std::stringstream myStream;

    myStream << var1 << ' ';
    myStream << std::fixed << std::setprecision(5) << var2;

    std::string var1Str, var2Str;
    myStream >> var1Str >> var2Str;

    std::cout << var1Str << ' ' << var2Str;

    return 0;
}
```

```
32767
5559.00001
```

# Конверсия число<->строка с помощью функций

```
int main() {
    std::string s1 = "12345.6777";
    std::string s2 = "ghghjf55.63";

    double d1 = std::stod(s1);
    std::cout << std::fixed << std::setprecision(4);
    std::cout << d1;

    double d2 = std::stod(s2);
    std::cout << d2;

    return 0;
}
```

# Конверсия число<->строка с помощью функций

```
int main() {
    std::string s1 = "12345.6777";
    std::string s2 = "ghghjf55.63";

    double d1 = std::stod(s1);
    std::cout << std::fixed << std::setprecision(4);
    std::cout << d1;

    double d2 = std::stod(s2);
    std::cout << d2;

    return 0;
}
```

- stoi( )
- stol( )
- stoll( )
- stof( )
- stod( )
- . . .

**Исключение!**