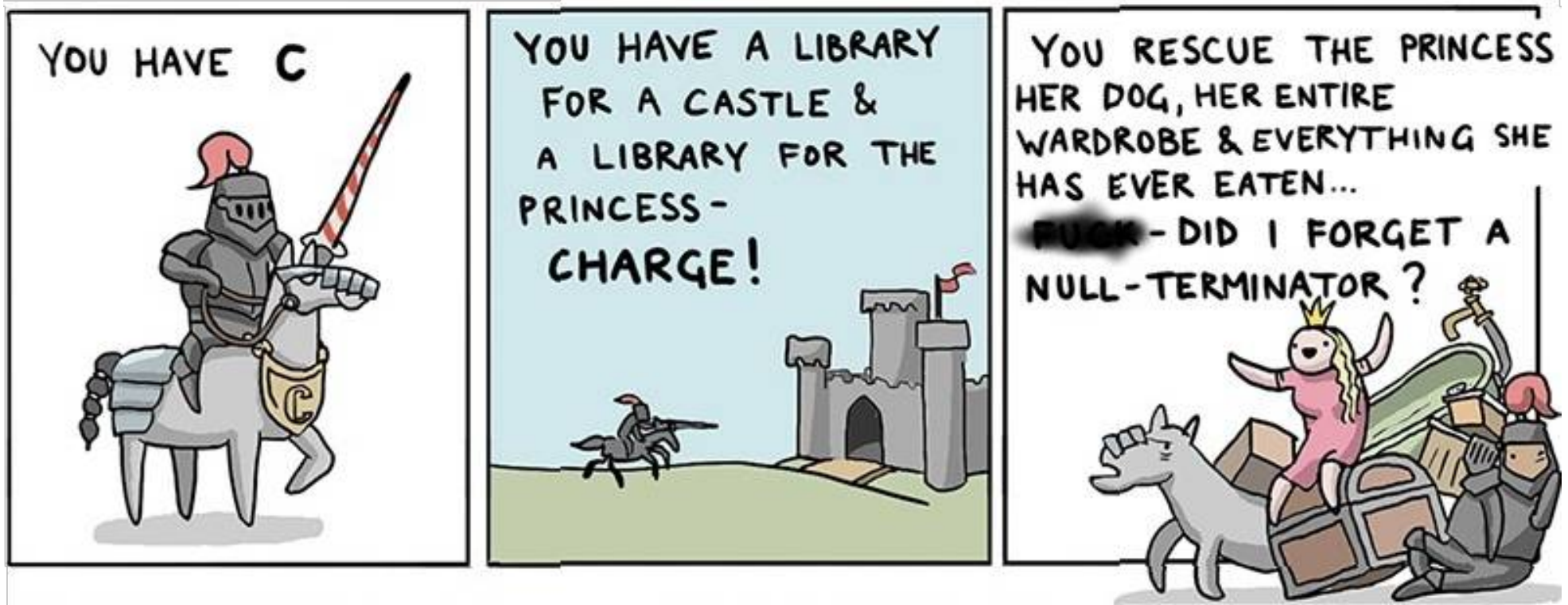


# Строки:С, С++ строки и Классификация алгоритмов на строках

Занятие 11 в ФМЛ 5  
г. Долгопрудного

# О спасении принцесс



# C Strings

Что значит `char *a="abcdef123";`

# C Strings

Что значит `char *a="abcdef123";`

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
a	b	c	d	e	f	1	2	3	\0

# C Strings

Что значит `char *a="abcdef123";`

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
a	b	c	d	e	f	1	2	3	\0

`a` — это указатель, находится в стеке,  
а сама строка — в data segment

# Data segments

Stack  
|  
v  
mmap  
|  
v  
^  
|  
heap

BSS — static uninitialized (zeroes)

Data segment — static initialized

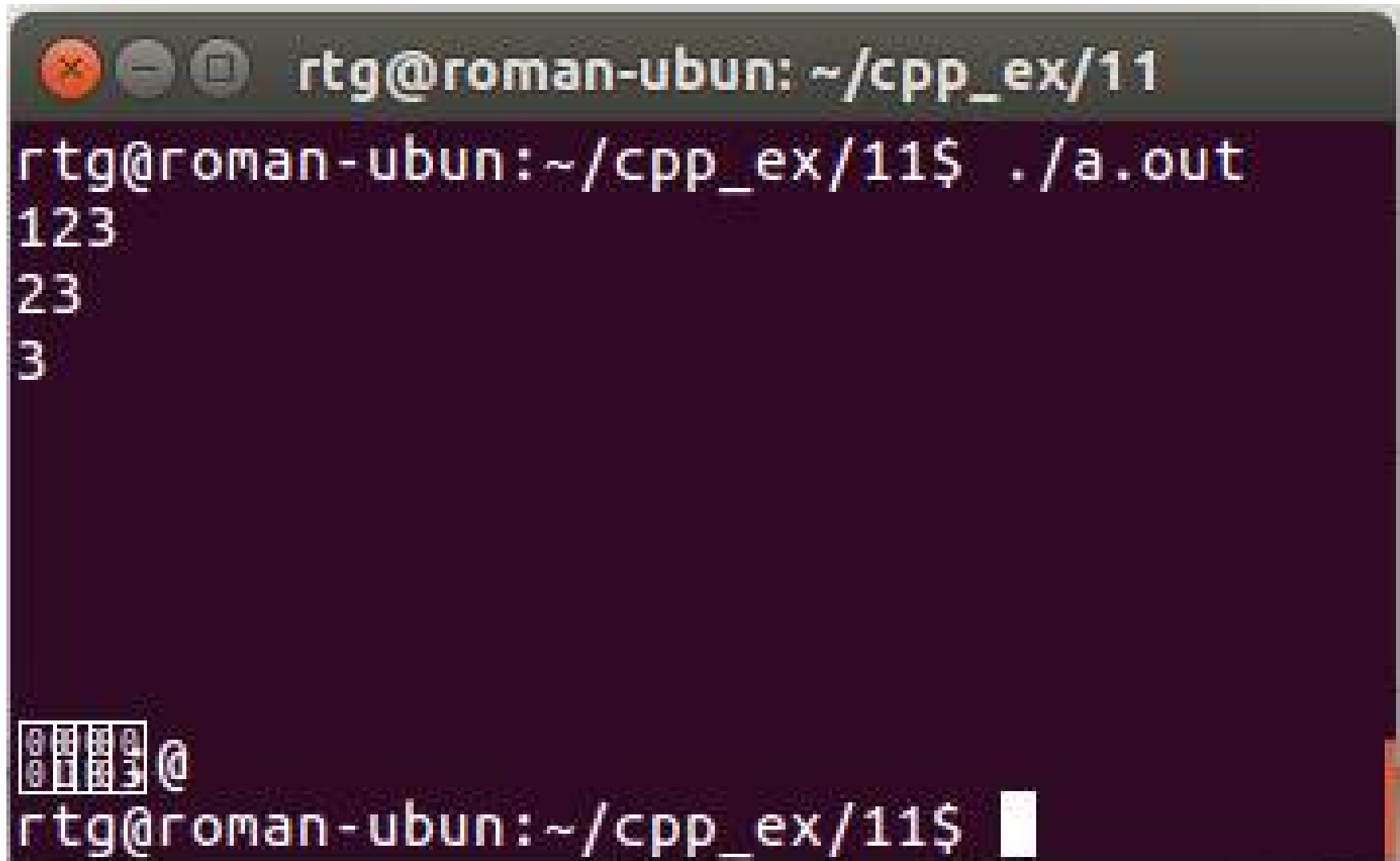
Text Segment (eIF) — binary data

# Пример

- `#include <iostream>`
- `#include <cstring>`
- 
- `int main(int argc, char **argv)`
- `{`
- `char *a = "123\0\0";`
- 
- `for(int i=0;i<7;i++)`
- `std::cout << a+i;`
- 
- `return 0;`
- `}`

# Вывод под Ubuntu

- В терминале

A screenshot of a terminal window on Ubuntu. The window title is 'rtg@roman-ubun: ~/cpp\_ex/11'. The prompt is 'rtg@roman-ubun:~/cpp\_ex/11\$'. The user has entered './a.out', and the program has output three lines: '123', '23', and '3'. At the bottom, there is a memory dump showing hexadecimal and ASCII values, followed by the prompt 'rtg@roman-ubun:~/cpp\_ex/11\$' and a cursor.

```
rtg@roman-ubun: ~/cpp_ex/11
rtg@roman-ubun:~/cpp_ex/11$ ./a.out
123
23
3
00000000 @
00000003 @
rtg@roman-ubun:~/cpp_ex/11$
```



# Операции

- strcpy
- `char *strcpy( char *restrict dest, const char *restrict src );` // (since C99)
- Не определена если получатель недостаточно большой

# Char

- Проверка символа:
- isalnum, isalpha, islower, isupper, isdigit, isxdigit, iscntrl, isgraph, isspace, isblank, isprint, ispunct
- Изменение символа:
- tolower, toupper

# Конвертация с-строк

`<stdlib.h>`

- `atof`
- `atoi atol atoll`
- `strtol strtoll`
- `strtoul strtoull`
- `strtod strtold`

`<inttypes.h>`

- `strtoimax strtoumax`

# Операции с строками

`<string.h>`

- `strlen strlen_s` — длина строки
- `strcmp` — сравнение
- `strncmp` — сравнение `n`
- `Strcoll` — сравнить с локалью
- `strchr` — найти букву
- `Strrchr` - найти последнюю букву
- `Strspn` — максимальная длина подстроки составленная из букв другой строки
- `Strcspn` - максимальная длина подстроки составленная не из букв другой строки
- `Strpbrk` — найти символы из строки 2 в строке 1
- `Strstr` — найти подстроку
- `strtok strtok_s` — найти токен в строке, записать `\0` в разделитель

# На соревнованиях / интервью

- Реализация быстрого варианта функции работы со строками

# C++ строки

- `std::string`    `std::basic_string<char>`
- `std::wstring`   `std::basic_string<wchar_t>`
- `std::u16string`   `std::basic_string<char16_t>`
- `std::u32string`   `std::basic_string<char32_t>`

# C++ строки: Сравнение

- Operator+
- Operator==
- operator!=
- operator<
- Operator>
- Operator<=
- operator>=

# С++ строки: Чтение/запись в поток

- Operator<<
- Operator >>



# С++ строки: Функции над строками

- `stoi stol stoll`
- `stoul stoull`
- `stof stod stold`
- `to_string`
- `to_wstring`
- `operator""s` (C++14) converts a character array literal to `basic_string`

# С++ строки: Члены класса строки: Доступ

- at
- operator[]
- front
- back
- data
- c\_str

# С++ строки: Итераторы и связанное

- `begin cbegin`
- `end cend`
- `rbegin crbegin`
- `rend crend`
- `capacity`
- `shrink_to_fit`

# C++ строки: Поиск

- `find`
- `rfind`
- `find_first_of`
- `find_first_not_of`
- `find_last_of`
- `find_last_not_of`

# C++ строки: операции

insert

erase,

clear => erase(begin(), end())

push\_back, pop\_back

append, operator+=

compare

replace

substr

copy

resize

swap

# На тренировку:

- Есть два массива строк. В каждом по 100000 строк. В каждой по 1000 букв (буквы в первом массиве формируются функцией `rand%32+32`) .
- Во втором массиве  $j$ -я строка отличается от  $i$ -й строки на позиции  $j \% 1000$
- Какое сравнение быстрее? `Strcmp(cstring1, cstring2)` или `string1 == string2`

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Поиск
- Сортировка
- Поиск расстояние
- Эффективное хранение (деревья)
- Паттерны и регулярные выражения
- Еще?

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Поиск
  - Наивный
  - Кнутта-Мориса-Пратта
  - Байеса-мура
  - На базе суффиксных деревьев
  - На базе других структур
  - Еще?



# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Сортировка

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Сортировка
  - Пузырьковая
  - Вставками
  - Слиянием
  - Быстрая
  - Пирамидальная
  - Карманная, поразрядная, сортировка подсчетом
  - Еще?

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Поиск расстояния
  - Расстояния Хемминга
  - Расстояния Левенштейна
  - LCS
  - Алгоритм Ханта-Шиманского ( $n \log n$ ,  $n$  памяти)
  - Еще?

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Сортировка
  - Пузырьковая
  - Вставками
  - Слиянием
  - Быстрая
  - Пирамидальная
  - Карманная, поразрядная, сортировка подсчетом
  - Еще?

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Эффективное хранение (деревья)
  - Суффиксные деревья
  - Префиксные
  - Можно использовать другие структуры
    - Такие как красно-черные деревья
    - Хэш-таблицы
    - В-деревья — (в базах данных)

# Классификация

Разберемся какие у нас есть, и какие нам нужны

- Паттерны и регулярные выражения
  - Какие интересны кроме скобок?
  - Как решаете?

# Что дальше

- Рассмотрим решение для Кнута — Морриса — Пратта на следующем занятии
- Ваши предложения — что делаем на занятии через одно. Варианты:
  - Разобрать код задачи про лягушонка
  - Код для задачи про Мемори и Соню
  - Еще чтонибудь?

# Вопросы и ответы

- Всем спасибо!