

# Библиотеки и оптимизации



Санкт-Петербургский  
государственный университет

# Про сдачу домашек

В курсе по Python вас научили пользоваться Git и GitHub

- Каждая домашняя работа оформляется как Pull Request в свой репозиторий
- Добавьте преподавателя в коллaborаторы
- Одно задание — один PR
- Следите за оформлением кода, коммитов, комментариев и т.д.

## Важно

Начиная с этой домашки в HwProj должна быть ссылка на Pull Request!

# Модули

- Способ группировки кода в логически обособленные группы
- В С это реализуется с помощью заголовочных файлов и файлов с реализацией
  - .h и .c
- В отдельный модуль выносятся объявления типов данных и функции, которые делают одно дело
  - Например, разные функции сортировки
  - Или всё для работы с матрицами
- В интерфейсную часть модуля выносится только то, что может использовать другой код
  - Меньше знаешь — крепче спиши
- Функции, используемые только для реализации, пишутся только в .c-файле
  - Например, функция разделения массива для быстрой сортировки или swap

# Модули

Заголовочный файл:

```
#pragma once
```

```
// Комментарий к функции 1  
int functionOne(int x, int y);
```

```
// Комментарий к функции 2  
void functionTwo();
```

.c-файл:

```
#include <имя заголовочного файла.h>
```

```
#include <все остальные библиотеки>
```

```
int functionOne(int x, int y)  
{  
    ...  
}
```

```
void functionTwo()  
{  
    ...  
}
```

# Тонкости

- Реализации функций в .h-файле писать нельзя
  - Иначе будет беда, если один .h-ник подключат в два .c-шника
- Комментарии обязательны
- `#pragma once` обязательна
- Подключать «свой» заголовочный файл в .c обязательно
- Файлы .h/.c всегда ходят парами, кроме файла с `main`

# Как это собрать

См. `workdir`

```
$ gcc -Wall -Wextra -pedantic main.c -o main  
/usr/bin/ld: /tmp/cc4YmGLo.o: in function `main':  
main.c:(.text+0x3f): undefined reference to `isPrime'  
collect2: error: ld returned 1 exit status
```

Не выйдет! Нужно так:

```
$ gcc -Wall -Wextra -pedantic prime.c -c  
$ gcc -Wall -Wextra -pedantic prime.o main.c -o main
```

# Библиотеки

- Многие задачи решали до вас неоднократно
- Такие готовые решения принято называть «библиотеками»
- Иногда требуется создавать свои библиотеки
- Чаще — достаточно использовать уже готовые

# Библиотеки

- Многие задачи решали до вас неоднократно
- Такие готовые решения принято называть «библиотеками»
- Иногда требуется создавать свои библиотеки
- Чаще — достаточно использовать уже готовые

## Определение

Библиотекой в Си называют набор заголовочных файлов, доступных другим приложениям, и файлов с реализацией объявлений из соответствующего заголовочного файла. Это могут быть как .c-файлы, так и уже прекомпилированные файлы (.lib, .dll, .so).

# Стандартные библиотеки в Си

На самом деле Вы использовали уже довольно много разных библиотек:

`stdio.h` `printf`, `scanf`, ...

`stdlib.h` `malloc`, `calloc`, `free`, ...

`math.h` `abs`, `sin`, `cos`, ...

`string.h` `strcmp`, `strcpy`, ...

`stdbool.h` `true`, `false`, ...

...

# Тонкости (для Linux)

- В системе библиотеки обычны живут в /usr/lib
- Заголовочные файлы в /usr/include
- При сборке библиотеки подключаются флагом -l<название библиотеки>
- Если библиотека находится по нестандартному пути, используйте флаг -L <путь>
- Если её заголовочные файлы находятся по нестандартному пути, используйте флаг -I <путь>

# Связывание с внешней библиотекой

На примере libcurl

```
$ curl https://raw.githubusercontent.com/curl/curl/master/docs/examples/https.c  
→ -o https.c  
$ gcc -Wall -Werror -pedantic https.c -o https # ошибка!  
$ gcc -Wall -Werror -pedantic https.c -lcurl -o https
```

# Статическая линковка

Способ линковки, при котором библиотеки встраиваются в исполняемый файл

Плюсы:

- Высокая переносимость
- Повышенная безопасность
- Скорость исполнения

Минусы:

- Увеличение размера исполняемого файла
- Сложность управления версиями библиотек

# Динамическая линковка

Способ связывания, при котором система загружает библиотеки по ходу исполнения программы

Плюсы:

- Меньший размер исполняемого файла
- «Простота» управления версиями библиотек

Минусы:

- Проблемы с переносимостью
- Временные затраты на поиск и загрузку библиотек по ходу исполнения

## Динамическая линковка vs. Статическая линковка

```
$ gcc -Wall -Wextra -pedantic root.c -lm -o root
$ ldd root
linux-vdso.so.1 (0x00007f4e00c64000)
libm.so.6 => /usr/lib/libm.so.6 (0x00007f4e00b16000)
libc.so.6 => /usr/lib/libc.so.6 (0x00007f4e00800000)
/lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2
    (0x00007f4e00c66000)
$ du -h root
16K  root
```

```
$ gcc -Wall -Wextra -pedantic root.c -lm -o root -static
$ ldd root
not a dynamic executable
$ du -h root
948K  root
```

# Компиляторные оптимизации

gcc -O1 -O2 -O3 ...

- раскрытие циклов (избавление от условных переходов)
- замена инструкций (умножение на 2 эквивалентно битовому сдвигу)
- раскрытие внутренних методов (избавление от вызовов подпрограмм)
- использование SSE (одновременная обработка блоков данных)
- ...

gcc -Os — оптимизация по размеру

Compiler Explorer: <https://godbolt.org/>

## Домашнее задание

Реализовать приложение, сортирующее поступающий в стандартный поток ввода набор целых чисел (не более 100 штук, это гарантируется). Числа разделены пробелами, последним символом является перенос строки.

Процедура сортировки должна быть реализована на языке ассемблера достаточно оптимальным образом. Кодом возврата приложения является количество элементов, участвовавших в сортировке и изменивших свою позицию.

Дополнительные соглашения:

- Чтение поступающих на вход сортируемых элементов необходимо производить с помощью команды `scanf(...)`.
- Файлы с кодом приложения должны иметь расширение `.c`, файл с реализацией сортировки на ассемблере должен иметь расширение `.s`.

Не забудьте приложить инструкцию по сборке!

# Полезные ссылки

- C FAQ: «I'm wondering what to put in .c files and what to put in .h files. (What does ".h" mean, anyway?)»
  - <https://c-faq.com/cpp/hfiles.html>
- How to handle dynamic and static libraries in Linux
  - <https://opensource.com/article/20/6/linux-libraries>
  - Статья для тех, кто хочет понять, как создавать библиотеки пригодные для связывания
- Про компоновку, dependency hell и обратную совместимость
  - <https://habr.com/ru/articles/220961/>
  - Про то, к каким побочным эффектам может приводить повсеместное бездумное использование динамическойリンクовки и как их можно лечить