Libft

Ваша первая собственная библиотека

Резюме:
Этот проект посвящен кодированию библиотеки на языке Си.
Он будет содержать множество функций общего назначения, на которые будут опираться ваши программы.

Версия: 15

Содержание

Глава I

Введение

Программирование на языке Си может быть очень утомительным, если не иметь доступа к очень полезным стандартным функциям. Этот проект посвящен пониманию того, как работают эти функции, их внедрению и обучению их использованию. Вы создадите свою собственную библиотеку. Она будет полезна, поскольку вы будете использовать ее в своих следующих школьных заданиях по Си.

Уделяйте время расширению своей библиотеки в течение года. Однако, работая над новым проектом, не забудьте убедиться, что функции, используемые в вашей библиотеке, разрешены руководством проекта.

Глава II

Общие инструкции

- Ваш проект должен быть написан на языке C.
- Ваш проект должен быть написан в соответствии с Нормой. Если у вас есть бонусные файлы/функции, они включаются в проверку нормы, и вы получите 0, если внутри есть ошибка нормы.
- Ваши функции не должны завершаться неожиданно (segmentation fault, bus error, double free, etc), кроме неопределенного поведения. Если это произойдет, ваш проект будет считаться не функциональным и получит 0 баллов во время оценки.
- Все выделенное пространство памяти должно быть надлежащим образом освобождено, когда это необходимо. Утечки не допускаются.
- Если тема требует этого, вы должны предоставить Makefile, который скомпилирует ваши исходные файлы до требуемого результата с флагами -Wall, -Wextra и -Werror, использовать cc, и ваш Makefile не должен перелинковываться.
- Ваш Makefile должен, по крайней мере, содержать правила $(NAME), all, clean, fclean и
  re.
- Чтобы внести бонусы в свой проект, вы должны включить в Makefile правило bonus, которое добавит все различные заголовки, либрейки или функции, запрещенные в основной части проекта. Бонусы должны находиться в другом файле _bonus.{c/h}, если в теме не указано ничего другого. Оценка обязательной и бонусной частей производится отдельно.
- Если ваш проект позволяет использовать свою libft, вы должны скопировать ее исходные тексты и связанный с ней Makefile в папку libft с ее связанным Makefile. Makefile вашего проекта должен скомпилировать библиотеку, используя ее Makefile, а затем скомпилировать проект.
- Мы рекомендуем вам создавать тестовые программы для вашего проекта, даже если эта работа **не будет сдаваться и не будет оцениваться**. Это даст вам возможность легко проверить свою работу и работу ваших коллег. Эти тесты будут особенно полезны во время защиты. Действительно, во время защиты вы можете использовать свои тесты и/или тесты коллеги, которого вы оцениваете.
- Отправьте свою работу в назначенный вам git-репозиторий. Оцениваться

будет только работа в git-репозитории. Если Deepthought назначит оценку вашей работы, это будет сделано

после оценки ваших работ сверстниками. Если во время оценивания Deepthought в каком-либо разделе вашей работы будет допущена ошибка, оценка будет остановлена.

Глава III

Обязательная часть

Название программы	libft.a
Сдать файлы	Makefile, libft.h, ft_*.c
Makefile	NAME, all, clean, fclean, re
Внешние функции.	Подробнее ниже
Либфт уполномочен	н/а
Описание	Напишите свою собственную библиотеку: набор функций который станет полезным инструментом для вашего курзуса.

III.1 Технические соображения

- Объявление глобальных переменных запрещено.
- Если вам нужны вспомогательные функции для разделения более сложной функции, определите их как статические функции. Таким образом, область их применения будет ограничена соответствующим файлом.
- Поместите все файлы в корень вашего хранилища.
- Сдача неиспользуемых файлов запрещена.
- Каждый .c файл должен компилироваться с флагами -Wall -Wextra -Werror.
- Вы должны использовать команду ar для создания библиотеки. Использование команды libtool запрещено.
- Ваш libft.a должен быть создан в корне вашего репозитория.

III.2 Часть 1 - Функции Libc

Для начала вы должны переделать набор функций из libc. Ваши функции будут иметь те же прототипы и реализовывать то же поведение, что и оригиналы. Они должны соответствовать тому, как они определены в man. Единственным отличием будут их имена. Они будут начинаться с префикса 'ft_'. Например, strlen станет ft_strlen.

Некоторые прототипы функций, которые вам придется переделать, используют квалификатор 'restrict'. Это ключевое слово является частью стандарта c99. Поэтому запрещается включать его в собственные прототипы и компилировать код с флагом -std=c99.

Вы должны написать свою собственную функцию, реализующую следующие исходные функции. Они не требуют никаких внешних функций:

- isalpha
- isdigit
- isalnum
- isascii
- isprint
- strlen
- memset
- bzero
- memcpy
- memmove
- strlcpy
- strlcat
- таппер
- ниже
- strchr
- strrchr
- strncmp
- memchr
- memcmp
- strnstr
- atoi

Для реализации двух следующих функций вы будете использовать malloc():

- calloc
- strdup

III.3 Часть 2 - Дополнительные функции

В этой второй части вы должны разработать набор функций, которые либо отсутствуют в libc, либо являются его частью, но в другой форме.

Некоторые из следующих функций могут быть полезны для написания функций части 1.

Название функции	ft_substr
Прототип	char *ft_substr(char const *s, unsigned int start, size_t len);
Сдать файлы	–
Параметры	s: Строка, из которой создается подстрока. начало: Начальный индекс подстроки в строке 's'. len:Максимальная длина подстроки.
Возвращаемое значение	Подстрока. NULL, если распределение не удалось.
Внешние функции.	malloc
Описание	Выделяет (с помощью malloc(3)) и возвращает подстроку из строки 's'. Подстрока начинается с индекса 'start' и имеет максимальный размер 'len'.

Название функции	ft_strjoin
Прототип	char *ft_strjoin(char const *s1, char const *s2);
Сдать файлы	–
Параметры	s1: Префиксная строка. s2: Суффиксная строка.
Возвращаемое значение	Новая строка. NULL, если распределение не удалось.
Внешние функции.	malloc
Описание	Выделяет (с помощью malloc(3)) и возвращает новый строка, которая является результатом конкатенации 's1' и 's2'.

Название функции	ft_strtrim
Прототип	char *ft_strtrim(char const *s1, char const *set);
Сдать файлы	–
Параметры	s1: Строка, которую нужно обрезать. набор: Эталонный набор символов для обрезки.
Возвращаемое значение	Обрезанная струна. NULL, если распределение не удалось.
Внешние функции.	malloc
Описание	Выделяет (с помощью malloc(3)) и возвращает копию 's1' с символами, указанными в 'set', удаленными из начала и конца строки.

Название функции	ft_split
Прототип	char **ft_split(char const *s, char c);
Сдать файлы	–
Параметры	s: Строка, которую нужно разделить. c: Символ-разделитель.
Возвращаемое значение	Массив новых строк, полученных в результате разбиения. NULL, если распределение не удалось.
Внешние функции.	malloc, free
Описание	Выделяет (с помощью malloc(3)) и возвращает массив строк, полученных путем разбиения 's' с использованием символа 'c' в качестве разделителя. Массив должен заканчиваться указателем NULL.

Название функции	ft_itoa
Прототип	char *ft_itoa(int n);
Сдать файлы	–
Параметры	n: целое число для преобразования.
Возвращаемое значение	Строка, представляющая целое число. NULL, если распределение не удалось.
Внешние функции.	malloc
Описание	Выделяет (с помощью malloc(3)) и возвращает строку представляющее целое число, полученное в качестве аргумента. Отрицательные числа должны обрабатываться.

Название функции	ft_strmapi
Прототип	char *ft_strmapi(char const *s, char (*f)(unsigned int, char));
Сдать файлы	-
Параметры	s: Строка, по которой выполняется итерация. f: Функция, применяемая к каждому символу.
Возвращаемое значение	Строка, созданная в результате последовательного применения 'f'. Возвращает NULL, если распределение не удалось.
Внешние функции.	malloc
Описание	Применяет функцию 'f' к каждому символу символа строку 's', и, передавая ее индекс в качестве первого аргумента, создать новую строку (с помощью malloc(3)), полученную в результате последовательных применений 'f'.

Название функции	ft_striteri
Прототип	void ft_striteri(char *s, void (*f)(unsigned int, char*));
Сдать файлы	-
Параметры	s: Строка, по которой выполняется итерация. f: Функция, применяемая к каждому символу.
Возвращаемое значение	Нет
Внешние функции.	Нет
Описание	Применяет функцию 'f' к каждому символу из числа строку, переданную в качестве аргумента, передавая ее индекс в качестве первого аргумента. Каждый символ передается по адресу в 'f' для изменения при необходимости.

Название функции	ft_putchar_fd
Прототип	void ft_putchar_fd(char c, int fd);
Сдать файлы	-
Параметры	c: Символ для вывода. fd: Дескриптор файла, на который производится запись.
Возвращаемое значение	Нет
Внешние функции.	написать

Описание	Выводит символ 'c' в заданный файл дескриптор.

Название функции	ft_putstr_fd
Прототип	void ft_putstr_fd(char *s, int fd);
Сдать файлы	–
Параметры	s: Строка для вывода. fd: Дескриптор файла, на который производится запись.
Возвращаемое значение	Нет
Внешние функции.	написать
Описание	Выводит строку 's' в заданный файл дескриптор.

Название функции	ft_putendl_fd
Прототип	void ft_putendl_fd(char *s, int fd);
Сдать файлы	–
Параметры	s: Строка для вывода. fd: Дескриптор файла, на который производится запись.
Возвращаемое значение	Нет
Внешние функции.	написать
Описание	Выводит строку 's' в заданный дескриптор файла за которым следует новая строка.

Название функции	ft_putnbr_fd
Прототип	void ft_putnbr_fd(int n, int fd);
Сдать файлы	–
Параметры	n: Целое число для вывода. fd: Дескриптор файла, на который производится запись.
Возвращаемое значение	Нет
Внешние функции.	написать
Описание	Выводит целое число 'n' в заданный файл дескриптор.

Глава IV

Бонусная часть

Если вы выполнили обязательную часть, не стесняйтесь пройти дальше, выполнив эту дополнительную часть. В случае успешного прохождения она принесет бонусные баллы.

Функции для манипулирования памятью и строками очень полезны. Но вскоре вы обнаружите, что манипулирование списками еще более полезно.

Вы должны использовать следующую структуру для представления узла вашего списка. Добавьте ее объявление в файл libft.h:

```
typedef struct      s_list
{
  void              *content;
  struct s_list     *next;
}                   t_list;
```

void * позволяет хранить данные любого типа.

- next: Адрес следующего узла, или NULL, если следующий узел является последним.

В Makefile добавьте правило make bonus, чтобы добавить бонусные функции в libft.a.

Бонусная часть оценивается только в том случае, если обязательная часть выполнена безупречно. Совершенство означает, что обязательная часть выполнена полностью и работает без сбоев. Если вы не выполнили ВСЕ обязательные требования, ваша бонусная часть не будет оцениваться вообще.

Реализуйте следующие функции, чтобы легко использовать ваши списки.

Название функции	ft_lstnew
Прототип	t_list *ft_lstnew(void *content);
Сдать файлы	–
Параметры	содержимое: Содержимое для создания узла.
Возвращаемое значение	Новый узел
Внешние функции.	malloc
Описание	Выделяет (с помощью malloc(3)) и возвращает новый узел. Переменная-член 'content' инициализируется значением параметра 'content'. Переменная 'next' инициализируется значением NULL.

Название функции	ft_lstadd_front
Прототип	void ft_lstadd_front(t_list **lst, t_list *new);
Сдать файлы	–
Параметры	lst: Адрес указателя на первую ссылку из список. новый: Адрес указателя на узел, который будет добавлен в список.
Возвращаемое значение	Нет
Внешние функции.	Нет
Описание	Добавляет узел 'new' в начало списка.

Название функции	ft_lstsize
Прототип	int ft_lstsize(t_list *lst);
Сдать файлы	–
Параметры	lst: Начало списка.
Возвращаемое значение	Длина списка
Внешние функции.	Нет
Описание	Подсчитывает количество узлов в списке.

Название функции	ft_lstlast
Прототип	t_list *ft_lstlast(t_list *lst);
Сдать файлы	–

Параметры	lst: Начало списка.
Возвращаемое значение	Последний узел списка
Внешние функции.	Нет
Описание	Возвращает последний узел списка.

Название функции	ft_lstadd_back
Прототип	void ft_lstadd_back(t_list **lst, t_list *new);
Сдать файлы	–
Параметры	lst: Адрес указателя на первую ссылку из список. новый: Адрес указателя на узел, который будет добавлен в список.
Возвращаемое значение	Нет
Внешние функции.	Нет
Описание	Добавляет узел 'new' в конец списка.

Название функции	ft_lstdelone
Прототип	void ft_lstdelone(t_list *lst, void (*del)(void *));
Сдать файлы	–
Параметры	lst: Узел, который нужно освободить. del: Адрес функции, используемой для удаления содержимого.
Возвращаемое значение	Нет
Внешние функции.	бесплатно
Описание	Принимает в качестве параметра узел и освобождает память от содержимое узла с помощью функции 'del', заданной в качестве параметра, и освободить узел. Память 'next' не должна быть освобождена.

Название функции	ft_lstclear
Прототип	void ft_lstclear(t_list **lst, void (*del)(void *));
Сдать файлы	–
Параметры	lst: Адрес указателя на узел. del: Адрес функции, используемой для удаления содержимого узла.
Возвращаемое значение	Нет
Внешние функции.	бесплатно
Описание	Удаляет и освобождает данный узел и каждый преемник этого узла, используя функцию 'del' и free(3). Наконец, указатель на список должен быть установлен в NULL.

Название функции	ft_lstiter
Прототип	void ft_lstiter(t_list *lst, void (*f)(void *));
Сдать файлы	–
Параметры	lst: Адрес указателя на узел. f: Адрес функции, используемой для итерации по списку.
Возвращаемое значение	Нет
Внешние функции.	Нет
Описание	Итерирует список 'lst' и применяет функцию 'f' на содержимое каждого узла.

Название функции	ft_lstmap
Прототип	t_list *ft_lstmap(t_list *lst, void *(*f)(void *), void (*del)(void *));
Сдать файлы	–
Параметры	lst: Адрес указателя на узел. f: Адрес функции, используемой для итерации по списку. del: Адрес функции, используемой для удаления содержимого узла, если это необходимо.
Возвращаемое значение	Новый список. NULL, если распределение не удалось.
Внешние функции.	malloc, free
Описание	Итерирует список 'lst' и применяет функцию 'f' на содержимом каждого узла. Создает новый список в результате последовательного применения функции 'f'. Функция 'del' используется для удаления содержимого узла, если это необходимо.

Глава V

Представление и экспертная оценка

Сдайте задание в свой Git-репозиторий, как обычно. Во время защиты будет оцениваться только работа, находящаяся в вашем репозитории. Не стесняйтесь дважды проверять имена файлов, чтобы убедиться в их правильности.

Поместите все файлы в корень вашего хранилища.