s21_string+

Реализация библиотеки string.h с дополнениями.

Contents

Preamble

 s21_string+

1942 год, поздний вечер, Блетчли-парк, рабочий стол Алана Тьюринга. Уже почти год группа умнейших математиков, лингвистов и любителей кроссвордов пытается решить сложнейшую задачку - расшифровать немецкую шифровальную машину «Энигма», коды для которой меняются каждый день, а количество возможных комбинаций примерно равно двум в 64 степени. Группе часто приходилось придумывать различные алгоритмы, и для удобства коммуникации и ведения протоколов был разработан специальный набор ключевых слов и синтаксис их использования, в точности похожий на хорошо известный в нашей вселенной язык Си. Бывают же совпадения, скажите! Одно неудобство было у работников Блетчли-парка - всю описанную этим языком последовательность
действий приходилось держать в голове.

Проходя мимо стола Тьюринга, вы замечаете листок с надписью: «Для обработки букв, знаков препинания, слов и предложений».

- Что это, Алан? - обратились вы к задумчивому молодому человеку у окна.

-- Это функции, которые упростят нам жизнь! Ты же знаешь, расшифровать Энигму методом грубого перебора..скорее уж я женюсь на Джоан, чем у нас это получится. Поэтому, кажется мне, нам необходимо продолжить анализировать тексты, искать закономерности, совпадения. А для этого придется придумывать и описывать различные алгоритмы, связанные с обработкой этого самого текста. Поэтому нам необходим ряд функций, которые помогут это делать. Их придумыванием я сейчас и занимаюсь.

- А придумываешь на нашем новом унифицированном средстве описания алгоритмов?

-- Да, именно на нем и придумываю. Где же ещё нам могут пригодиться эти функции? - сказав это Тьюринг посмотрел на вас, как на недалёкого. Вы это поняли и решили блеснуть своей подкованностью в вопросе:

- Знаешь, мне кажется это нам действительно необходимо. Я как раз недавно выучил этот, назовем его так, специфический язык передачи алгоритмов.

-- Серьезно? - с некой заинтересованностью спросил Алан.

- Ну да.

Подумав пару секунд, Тьюринг решил, что будет логично доверить эту работу вам:

-- Знаешь, а хочешь сам заняться этим? Собери несколько не очень занятых людей и вперед. А я продолжу работу над своей механической машиной для переборки кодов.

Подумав пару секунд, вы решили, что это отличная идея:

- Да, сделаем все в лучшем виде!

Chapter I

Introduction

В данном проекте Вам предстоит разработать собственную реализацию библиотеки string.h на языке программирования Си с некоторыми дополнениями (с собственной реализацией функций sprintf и sscanf). Библиотека string.h является основной библиотекой языка Си по обработке строк. В рамках этого проекта предполагается отработка задач на работу со строковыми данными и закрепление структурного подхода.

Chapter II

Information

Язык программирования C содержит набор функций, реализующих операции со строками (символьными строками и строками байтов) в своей стандартной библиотеке. В ней поддерживаются различные операции, такие как копирование, конкатенация, маркировка и поиск. Для символьных строк в стандартной библиотеке существует правило о том, что строки должны заканчиваться терминирующим нуль-символом: строка из n символов представляется в виде массива из n + 1 элементов, последний из которых является символом "NULL".
Единственная поддержка строк собственно в языке программирования C заключается в том, что компилятор преобразует строковые константы в кавычках в строки, заканчивающиеся нулем.

string.h Типы

№	Переменная	Описание
1	size_t	Целочисленный тип без знака, являющийся результатом ключевого слова sizeof.

string.h Макросы

№	Макрос	Описание
1	NULL	Макрос, являющийся значением константы нулевого указателя.

string.h Функции

№	Функция	Описание

1	void *memchr(const void *str, int c, size_t n)	Выполняет поиск первого вхождения символа c (беззнаковый тип) в первых n байтах строки, на которую указывает аргумент str.
2	int memcmp(const void *str1, const void *str2, size_t n)	Сравнивает первые n байтов str1 и str2.
3	void *memcpy(void *dest, const void *src, size_t n)	Копирует n символов из src в dest.
4	void *memmove(void *dest, const void *src, size_t n)	Еще одна функция для копирования n символов из src в dest.
5	void *memset(void *str, int c, size_t n)	Копирует символ c (беззнаковый тип) в первые n символов строки, на которую указывает аргумент str.
6	char *strcat(char *dest, const char *src)	Добавляет строку, на которую указывает src, в конец строки, на которую указывает dest.
7	char *strncat(char *dest, const char *src, size_t n)	Добавляет строку, на которую указывает src, в конец строки, на которую указывает dest, длиной до n символов.
8	char *strchr(const char *str, int c)	Выполняет поиск первого вхождения символа c (беззнаковый тип) в строке, на которую указывает аргумент str.
9	int strcmp(const char *str1, const char *str2)	Сравнивает строку, на которую указывает str1, со строкой, на которую указывает str2.
10	int strncmp(const char *str1, const char	Сравнивает не более первых n байтов str1 и str2.

	*str2, size_t n)	
11	char *strcpy(char *dest, const char *src)	Копирует строку, на которую указывает src, в dest.
12	char *strncpy(char *dest, const char *src, size_t n)	Копирует до n символов из строки, на которую указывает src, в dest.
13	size_t strcspn(const char *str1, const char *str2)	Вычисляет длину начального сегмента str1, который полностью состоит из символов, не входящих в str2.
14	char *strerror(int errnum)	Выполняет поиск во внутреннем массиве номера ошибки errnum и возвращает указатель на строку с сообщением об ошибке. Нужно объявить макросы, содержащие массивы сообщений об ошибке для операционных систем mac и linux. Описания ошибок есть в оригинальной библиотеке. Проверка текущей ОС осуществляется с помощью директив.
15	size_t strlen(const char *str)	Вычисляет длину строки str, не включая завершающий нулевой символ.
16	char *strpbrk(const char *str1, const char *str2)	Находит первый символ в строке str1, который соответствует любому символу, указанному в str2.
17	char *strrchr(const char *str, int c)	Выполняет поиск последнего вхождения символа c (беззнаковый тип) в строке, на которую указывает аргумент str.
18	size_t strspn(const char *str1, const char *str2)	Вычисляет длину начального сегмента str1, который полностью состоит из символов str2.
19	char *strstr(const char *haystack, const char *needle)	Находит первое вхождение всей строки needle (не включая завершающий нулевой символ), которая появляется в строке haystack.
20	char	Разбивает строку str на ряд токенов, разделенных delim.

	*strtok(char *str, const char *delim)	

sprintf and sscanf

- int sscanf(const char *str, const char *format, ...) - считывает форматированный ввод из строки.
- int sprintf(char *str, const char *format, ...) - отправляет форматированный вывод в строку, на которую указывает str.

где:

- str − Это C-строка, которую функция обрабатывает в качестве источника для извлечения данных;
- format − это C-строка, содержащая один или несколько следующих элементов: пробельный символ, непробельный символ и спецификаторы формата. Спецификатор формата для печатающих функций следует прототипу: %[флаги][ширина][.точность][длина]спецификатор. Спецификатор формата для сканирующих функций следует прототипу: %[*][ширина][длина]спецификатор.

sprintf and sscanf Спецификаторы

№	Спецификатор	Результат sprintf	Результат sscanf
1	c	Символ	Символ
2	d	Знаковое десятичное целое число	Знаковое десятичное целое число
3	i	Знаковое десятичное целое число	Знаковое целое число (может быть десятичным, восьмеричным или шестнадцатеричным)
4	e	Научная нотация (мантисса/экспонента) с использованием символа e (вывод чисел должен совпадать с точностью до e-6)	Десятичное число с плавающей точкой или научная нотация (мантисса/экспонента)
5	E	Научная нотация (мантисса/экспонента) с использованием символа E	Десятичное число с плавающей точкой или научная нотация (мантисса/экспонента)
6	f	Десятичное число с плавающей точкой	Десятичное число с плавающей точкой или научная нотация (мантисса/экспонента)
7	g	Использует кратчайший из представлений десятичного числа	Десятичное число с плавающей точкой или научная нотация (мантисса/экспонента)
8	G	Использует кратчайший из представлений десятичного числа	Десятичное число с плавающей точкой или

			научная нотация (мантисса/экспонента)
9	o	Беззнаковое восьмеричное число	Беззнаковое восьмеричное число
10	s	Строка символов	Строка символов
11	u	Беззнаковое десятичное целое число	Беззнаковое десятичное целое число
12	x	Беззнаковое шестнадцатеричное целое число	Беззнаковое шестнадцатеричное целое число (любые буквы)
13	X	Беззнаковое шестнадцатеричное целое число (заглавные буквы)	Беззнаковое шестнадцатеричное целое число (любые буквы)
14	p	Адрес указателя	Адрес указателя
15	n	Количество символов, напечатанных до появления %n	Количество символов, считанных до появления %n
16	%	Символ %	Символ %

sprintf Флаги

№	Флаг	Описание
1	-	Выравнивание по левому краю в пределах заданной ширины поля; Выравнивание по правому краю используется по умолчанию (см. подспецификатор ширины).
2	+	Заставляет явно указывать знак плюс или минус (+ или -) даже для положительных чисел. По умолчанию только отрицательным числам предшествует знак "-".
3	(пробел)	Если знак не будет выведен, перед значением вставляется пробел.
4	#	При использовании со спецификаторами o, x или X перед числом вставляется 0, 0x или 0X соответственно (для значений, отличных от нуля). При использовании с e, E и f "заставляет" записанный вывод содержать десятичную точку, даже если за ней не последует никаких цифр. По умолчанию, если не следует никаких цифр, десятичная точка не записывается. При использовании с g или G результат такой же, как и с e или E, но конечные нули не удаляются.
5	0	Заполняет число слева нулями (0) вместо пробелов, где указан спецификатор ширины (см. подспецификатор ширины).

sprintf and sscanf Ширина

№	Ширина	Описание
1	(число)	Минимальное количество печатаемых символов. Если выводимое значение

		короче этого числа, результат дополняется пробелами. Значение не усекается, даже если результат больше.
2	*	В sprintf знак * значит, что ширина указывается не в строке формата, а в качестве дополнительного аргумента целочисленного значения, предшествующего аргументу, который необходимо отформатировать. В sscanf знак *, помещенный после % и перед спецификатором формата, считывает данные указанного типа, но подавляет их присваивание.

sprintf Точность

№	.точность	Описание
1	.число	Для целочисленных спецификаторов (d, i, o, u, x, X) – точность определяет минимальное количество записываемых цифр. Если записываемое значение короче этого числа, результат дополняется ведущими нулями. Значение не усекается, даже если результат длиннее. Точность 0 означает, что для значения 0 не записывается ни одного символа. Для спецификаторов e, E и f – это количество цифр, которые должны быть напечатаны после десятичной точки. Для спецификаторов g и G – это максимальное количество значащих цифр, которые должны быть напечатаны. Для s – это максимальное количество печатаемых символов. По умолчанию все символы печатаются до тех пор, пока не встретится терминирующий нуль. Для типа c – никак не влияет. Если точность не указана для спецификаторов e, E, f, g и G, то по умолчанию ее значение равно 6. Если точность не указана для остальных спецификаторов, то по умолчанию ее значение равно 1. Если число не указано (нет явного значения точности), то по умолчанию - 0.
2	.*	Точность указывается не в строке формата, а в качестве дополнительного аргумента целочисленного значения, предшествующего аргументу, который должен быть отформатирован.

sprintf and sscanf Длина

№	Длина	Описание
1	h	Аргумент интерпретируется как короткое int или короткое int без знака (применяется только к целочисленным спецификаторам: i, d, o, u, x и X).
2	l	Аргумент интерпретируется как длинное int или длинное int без знака для целочисленных спецификаторов (i, d, o, u, x и X) и как широкий символ или строка широких символов для спецификаторов c и s.
3	L	Аргумент интерпретируется как длинный double (применяется только к спецификаторам с плавающей точкой – e, E, f, g и G).

Специальные функции обработки строк (вдохновленные классом String в языке C#)

№	Функция	Описание

1	void *to_upper(const char *str)	Возвращает копию строки (str), преобразованной в верхний регистр. В случае какой-либо ошибки следует вернуть значение NULL
2	void *to_lower(const char *str)	Возвращает копию строки (str), преобразованной в нижний регистр. В случае какой-либо ошибки следует вернуть значение NULL
3	void *insert(const char *src, const char *str, size_t start_index)	Возвращает новую строку, в которой указанная строка (str) вставлена в указанную позицию (start_index) в данной строке (src). В случае какой-либо ошибки следует вернуть значение NULL
4	void *trim(const char *src, const char *trim_chars)	Возвращает новую строку, в которой удаляются все начальные и конечные вхождения набора заданных символов (trim_chars) из данной строки (src). В случае какой-либо ошибки следует вернуть значение NULL

Chapter III

Part 1. Реализация функций библиотеки string.h

Необходимо реализовать описанные выше функции библиотеки string.h:

- Библиотека должна быть разработана на языке Си стандарта C11 с использованием компилятора gcc
- Код библиотеки, включая заголовочные файлы, мейкфайлы и сама библиотека должны находиться в папке src в ветке develop
- Не использовать устаревшие и выведенные из употребления конструкции языка и библиотечные функции. Обращать внимания на пометки legacy и obsolete в официальной документации по языку и используемым библиотекам. Ориентироваться на стандарт POSIX.1-2017
- При написании кода необходимо придерживаться Google Style
- Оформить решение как статическую библиотеку (с заголовочным файлом s21_string.h)
- Библиотека должна быть разработана в соответствии с принципами структурного программирования, должно быть исключено дублирование в коде
- Подготовить полное покрытие unit-тестами функций библиотеки с помощью библиотеки Check
- Unit-тесты должны проверять результаты работы вашей реализации путём сравнения ее с реализацией стандартной библиотеки string.h
- Unit-тесты должны покрывать не менее 80% каждой функции
- Предусмотреть Makefile для сборки библиотеки и тестов (с целями all, clean, test, s21_string.a, gcov_report)
- В цели gcov_report должен формироваться отчёт gcov в виде html страницы. Для этого unit-тесты должны запускаться с флагами gcov
- Перед каждой функцией использовать префикс s21_
- Запрещено копирование реализации и использование стандартной библиотеки string.h и других библиотек по обработке строк везде, кроме unit-тестов
- Запрещено использование системных списков ошибок, включая списки, непрописанные в стандартах POSIX (sys_nerr, sys_errlist). Вместо этого необходимо реализовать

свои платформозависимые списки ошибок, как это было упомянуто в описании функции strerror
- Необходимо соблюсти логику работы стандартной библиотеки string.h (в части проверок, работы с памятью и поведения в нештатных ситуациях - здесь помогут тесты)
- Функции должны работать с z-строками из однобайтовых символов в кодировке ASCII.

Part 2. Частичная реализация функции sprintf

Необходимо реализовать функцию sprintf из библиотеки stdio.h:

- Функция должна быть размещена в библиотеке s21_string.h.
- На реализацию функции накладываются все требования, изложенные в первой части.
- Должно поддерживаться частичное форматирование:
  - Спецификаторы: c, d, i, f, s, u, %
  - Флаги: -, +, (пробел)
  - Ширина: (число)
  - Точность: .(число)
  - Длина: h, l

Part 3. Дополнительно. Реализация некоторых модификаторов формата функции sprintf

Необязательное задание на дополнительные баллы. Необходимо реализовать некоторые модификаторы формата функции sprintf из библиотеки stdio.h:

- Функция должна быть размещена в библиотеке s21_string.h.
- На реализацию функции накладываются все требования, изложенные в первой части.
- Должны поддерживаться следующие дополнительные модификаторы формата:
  - Спецификаторы: g, G, e, E, x, X, o, n, p
  - Флаги: #, 0
  - Ширина: *
  - Точность: .*
  - Длина: L

Part 4. Дополнительно. Реализация функции sscanf

Необязательное задание на дополнительные баллы. Необходимо реализовать функцию sscanf из библиотеки stdio.h:

- Функция должна быть размещена в библиотеке s21_string.h.
- На реализацию функции накладываются все требования, изложенные в первой части.
- Должно поддерживаться полное форматирование (с учетом флагов, ширины, точности, модификаторов и типов преобразования).

Part 5. Дополнительно. Реализация специальных функций обработки строк

Необязательное задание на дополнительные баллы. Необходимо реализовать некоторые функции обработки строк из класса String (описанные здесь):

- Функции должны быть размещены в библиотеке s21_string.h.

- На реализацию функций накладываются все требования, изложенные в первой части, исключая требование о сравнении вашей реализации со стандартом.

 Нажми тут, **чтобы поделиться с нами обратной связью на этот проект**. Это анонимно и поможет команде Педаго сделать твоё обучение лучше.