FASM_String library

Документация.

The Bird of Hermes, 2025 г.

Оглавление

1. Структура String (FASM_String_t)

FASM:

```
struct String
   __size  dd ?
   len     dd ?
   union
      __lpChars   dptr  ?
      __chars     db pointer.size dup(?)
   ends
ends
```

C:

```
typedef struct {
   int __size;
   int len;
   union{
    char __chars[sizeof(char *)];
    char *__lpChars;
   };
} FASM_String_t ;
```

Поля структуры

1.1. int __size

Хранит размер памяти, выделенной под структуру. Пользователь не должен получать или изменять это значение. При инициализации должен быть установлен в 0 или sizeof(void *) – 1.

1.2. int len

Хранит длину строки. Допустимо чтение этого поля, изменения недопустимы. Должно быть изначально инициализировано в 0.

1.3. char __chars[sizeof(char *)]

Может выступать как хранилище символов строки, если длина строки не превышает sizeof(void *) – 1.

1.4. char *__lpChars

Указатель на память, выделенную для строки.

Если изначальные поля структуры не заполнены, для первичной инициализации может использоваться функция String_alloc(метод String:alloc для FASM_OOP).

Класс FASM_String в C++ реализует доступ к полю len через метод FASM_String::len(). Этот класс является более высокоуровневой оберткой.

2. Функции структуры String (FASM_String_t)

2.1. String_print.

Вывод в консоль.

Сигнатура

void String_print(FASM_String_t *_this);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которую необходимо вывести.

Возвращаемое значение: нет.

Аналог в FASM_OOP версии – инлайн-метод(макрос) String:print. Аналог в C++ - метод FASM_String::print.

2.2. String_input.

Ввод строки с клавиатуры. Длина «не ограничена».

Сигнатура

void String_input(FASM_String_t *_this);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая получит введенные символы.

Возвращаемое значение: нет.

Аналог в FASM_OOP версии –метод String:input. Аналог в C++ - метод FASM_String::input.

2.3. String_getLpChars

Получает указатель на массив символов текущей строки. Пользователь не должен освобождать эту память. После вызова любой из функций строк может стать недействительным. Пользователь не должен освобождать эту память.

Сигнатура

char * String_getLpChars(FASM_String_t *_this);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, чей массив символов будет получен.

Возвращаемое значение: указатель на массив символов данной строки.

Аналог в FASM_OOP версии – инлайн-метод(макрос) String:getLpChars. Аналог в C++ - метод FASM_String::getLpChars. Так же используется перегруженным оператором индекса ([]) с числовым аргументом для получения символа по индексу.

2.4. String_flush

Сигнатура

char * String_flush(FASM_String_t *_this);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая будет очищена.

Возвращаемое значение: указатель на массив символов данной строки.

Очищает строку. Устанавливает значение длины строки в 0 и обнуляет начало массива байт. Эта функция НЕ ОСВОБОЖДАЕТ ПАМЯТЬ, занимаемую строкой.

Аналог в FASM_OOP версии – инлайн-метод(макрос) String:flush. Аналог в C++ - метод FASM_String::flush.

2.5. String_free

Сигнатура

void String_free(FASM_String_t *_this);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая будет освобождена.

Возвращаемое значение: нет.

Эта функция освобождает память, занятую строкой, если она была выделена. Поле __size получает значение sizeof(void *) – 1, поля len и __lpChars зануляются.

Аналог в FASM_OOP версии – метод String:free. Аналог в C++ - String::free, так же вызывается в деструкторе.

2.6. String_alloc

Сигнатура

void String_alloc(FASM_String_t *_this, int len);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, для которой выделяется память.

2. int len

   Количество байт памяти, резервируемых для строки.

Возвращаемое значение: нет.

Резервирует память для строки. Пользователь не должен вызывать её для строки, которая использовалась ранее и не была освобождена. Можно использовать для первичной инициализации. Допустимо использовать либо до любых других функций, работающих с данной строкой, либо после Strinng_free (для C++ - либо после конструктора без аргументов, либо после FASM_String::free, либо после деструктора). Если память ранее уже была выделена либо со строкой выполнялись какие-либо операции, используйте String_realloc.

Аналог в FASM_OOP – метод String:alloc. Аналог в C++ - метод FASM_String::alloc.

2.7. String_realloc

Сигнатура

void String_realloc(FASM_String_t *_this, int len);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, для которой перевыделяется память.

2. int len

   Новое количество байт памяти, резервируемых для строки.

Возвращаемое значение: нет.

Эта функция изменяет размер блока памяти, выделенного для строки.

Аналог в FASM_OOP – метод String:realloc. Аналог в C++ - метод FASM_String::realloc.

2.8. String_fromBytes

Сигнатура

void String_fromBytes(FASM_String_t *_this, char * bytes, int bCount);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. char * bytes

   Указатель на массив байт, из которых будет составлена строка.

3. int bCount

   Длина массива байт. Если значение -1, значит ASCIIZ строка.

Возвращаемое значение: нет.

Создает строку на основе массива байт.

Аналог в FASM_OOP – метод String:fromBytes. Аналог в C++ - конструктор класса с соответствующими параметрами (кроме первого), одна из перегрузок метода FASM_String::from или одна из перегрузок оператора присвоения(=).

2.9. String_addBytes

Сигнатура

void String_addBytes(FASM_String_t *_this, char * bytes, int bCount);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, к которой добавляется массив байт.

2. char * bytes

   Указатель на массив байт, который будет присоединен.

3. int bCount

   Длина массива байт. Если значение -1, значит ASCIIZ строка.

Возвращаемое значение: нет.

Создает строку на основе массива байт.

Аналог в FASM_OOP – метод String:addBytes. Аналог в C++ - одна из перегрузок метода FASM_String::add или перегрузка одного из операторов сложения(+, +=).

2.10. String_fromByte

Сигнатура

void String_fromByte(FASM_String_t *_this, char bVal, int bCount);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. char bVal

   Значение байта, которым заполнится строка.

3. int bCount

   Количество повторений байта, которое должно быть в итоговой строке.

Возвращаемое значение: нет.

Создает строку на основе N повторений байта.

Аналог в FASM_OOP – метод String:fromByte. Аналог в C++ - конструктор класса с соответствующими параметрами (кроме первого), одна из перегрузок метода FASM_String::from или одна из перегрузок оператора присвоения(=).

2.11. String_addByte

Сигнатура

void String_addByte(FASM_String_t *_this, char bVal, int bCount);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, к которой добавляется массив байт.

2. char bVal

   Значение байта, которым заполнится строка.

3. int bCount

   Количество повторений байта, которое должно быть добавлено к итоговой строке.

Возвращаемое значение: нет.

Добавляет в строку N повторений байта. Не подходит для первичной инициализации.

Аналог в FASM_OOP – метод String:addBytes. Аналог в C++ - одна из перегрузок метода FASM_String::add или перегрузка одного из операторов сложения(+, +=).

2.12. String_fromString

Сигнатура

void String_fromByte(FASM_String_t *_this, FASM_String_t *src);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. FASM_String_t *src

   Строка-источник.

Возвращаемое значение: нет.

Создает строку на основе другой строки.

Аналог в FASM_OOP – метод String:fromString: Аналог в C++ - конструктор класса с соответствующими параметрами (кроме первого), одна из перегрузок метода FASM_String::from или одна из перегрузок оператора присвоения(=).

2.13. String_addString

Сигнатура

void String_addByte(FASM_String_t *_this, FASM_String_t *src);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, к которой добавляется массив байт.

2. FASM_String_t *src

   Строка-источник.

Возвращаемое значение: нет.

Добавляет в строку другую строку.

Аналог в FASM_OOP – метод String:addString: Аналог в C++ - одна из перегрузок метода FASM_String::add или перегрузка одного из операторов сложения(+, +=).

2.14. String_fromNum

Сигнатура

void String_fromByte(FASM_String_t *_this, int num, int radix, int isSigned);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. int num

   Значение числа, которым заполнится строка.

3. int radix

   Основание системы счисления (от 2 до 36).

4. int isSigned

   Определяет, является ли число знаковым. При ненулевом значении параметра – знаковое, иначе – беззнаковое.

Возвращаемое значение: нет.

Создает строку из числа.

Аналог в FASM_OOP – метод String:fromNum. Аналог в C++ - конструктор класса с соответствующими параметрами (кроме первого), одна из

перегрузок метода FASM_String::from или одна из перегрузок оператора присвоения(=).

2.15. String_addNum

Сигнатура

void String_addNum(FASM_String_t *_this, int num, int radix, int isSigned);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. int num

   Значение числа, которым заполнится строка.

3. int radix

   Основание системы счисления (от 2 до 36).

4. int isSigned

   Определяет, является ли число знаковым. При ненулевом значении параметра – знаковое, иначе – беззнаковое.

Возвращаемое значение: нет.

Добавляет в строку число.

Аналог в FASM_OOP – метод String:addBytes. Аналог в C++ - одна из перегрузок метода FASM_String::add или перегрузка одного из операторов сложения(+, +=), которые так же определены для беззнаковых чисел.

2.16. String_slice

Сигнатура

void String_slice(FASM_String_t *_this, FASM_String_t *src, int _start, int_end);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. FASM_String_t *src

   Строка-источник.

3. int _start

   Индекс начала среза. Если индекс отрицательный, то отсчитывается с конца строки. Если при отсчете от конца все еще отрицательный, обрезается до 0.

4. int _end

   Индекс конца среза. Не включается. Если индекс отрицательный, отсчитывается с конца строки. Если индекс выходит за пределы длины строки, он обрезается до длины строки. Для среза до конца укажите длину строки или число, большее её

Возвращаемое значение: нет.

Создает строку на основе среза другой строки.

Аналог в FASM_OOP – метод String:slice. Аналог в C++ - метод FASM_String::slice, так же перегружен оператор индекса ([]) с параметром {int, int}, который возвращает объект-срез.

2.17. String_sprintf

Сигнатура

void String_sprintf(FASM_String_t *_this, fstr *format, ...);

Аргументы:

1. FASM_String_t *_this

   Указатель на строку, которая заполняется.

2. fstr *format

   Массив байт формата.

3. …

   Параметры для заполнения по строке формата.

Возвращаемое значение: нет.

Создает строку на основе формат-строки и переменного числа аргументов.

Аналог в FASM_OOP – метод String:sprintf. Аналог в C++ - метод FASM_String::sprint, имеющий 3 перегрузки для параметра строки формата – для класса FASM_Fstr, типа fstr и char *. Подробнее о формате sprintf – в следующем разделе.

3. Формат sprintf. Тип fstr *

3.1. Формат sprint

Спецификация преобразования состоит из необязательных и обязательных полей, имеющих следующий вид:

%[flags][width][.precision]type

Каждое поле спецификации преобразования — это символ или число, указывающее конкретный параметр формата или описатель преобразования. Обязательное поле *type* определяет тип преобразования, которое применяется к аргументу. Необязательные *флаги, ширина и точность полей определяют другие аспекты формата, такие как начальные пробелы* или нули, обоснование и отображаемая точность.

1. Тип аргумента

Символ спецификации преобразования *type* определяет, как должен интерпретироваться соответствующий аргумент: как символ, строка, указатель, целое число или число с плавающей запятой. Символ *type* — единственное обязательное поле спецификации преобразования; он указывается после всех необязательных полей.

Таблица 1. Типы

Символ	Аргумент	Формат вывода
c	символ	Однобайтовый символ
i, d	целое число	Десятичное целое число со знаком
u	целое число	Десятичное целое число без знака
x	целое число	Шестнадцатиричное целое число без знака
o	целое число	Восьмеричное целое число без знака
s	строка	Строка однобайтовых символов. Символы отображаются до первого нулевого символа или до тех пор, пока не будет достигнуто значение *precision*.
S	строка	Указатель на String(FASM_String_t) или обьект класса FASM_String: Символы отображаются до первого нулевого символа или до тех пор, пока не будет достигнуто значение *precision*.
%	нет	Добавляет символ % на свое место.

2. Флаги

Таблица 2. Значения флагов

Флаг	Значение	По умолчанию
«-»	Выравнивание результата по левому краю в пределах заданной ширины поля.	Выравнивание по правому краю.

«+»	Используется знак (+ или -), чтобы префиксировать выходное значение, если оно имеет числовой знаковый тип.	Знак отображается только для отрицательных значений со знаком –.
«0»	Не используется (добавлено для возможной поддержки чисел с плавающей точкой в будущем)	-
« »	То же, что и «+», но вместо + для положительных чисел будет использоваться пробел.	Пробел для положительных чисел не отображается.

3. Ширина

Неотрицательное целое число, которое управляет минимальным числом символов, которые являются выходными. Если количество символов в выходном значении меньше заданной ширины, к значениям слева или справа (в зависимости от того, определен ли флаг выравнивания по левому краю (-)) добавляются пробелы, в количестве, необходимом для достижения минимальной ширины.

Спецификация ширины никогда не вызывает усечения значения. Если число символов в выходном значении больше указанной ширины или если width оно не указано, все символы значения являются выходными, при условии, что его не обрежет параметр точности.

Если в качестве спецификации ширины указана звездочка (*), значение ширины задается аргументом int из списка аргументов. Аргумент width должен предшествовать значению, которое необходимо отформатировать, в списке аргументов.

4. Точность

В спецификации преобразования третье необязательное поле является спецификацией точности. Он состоит из периода (.), за которым следует неотрицательное целое число, значение которого зависит от типа преобразования.
Если спецификация точности представляет собой звездочку (*), аргумент int из списка аргументов предоставляет значение. В списке аргументов аргумент *precision* должен предшествовать форматируемому значению

Таблица 3. Влияние точности на тип

Тип	Значение	По умолчанию.
c	Кол-во повторений символа	1
d, i, u, o, x	Точность определяет минимальное выводимое количество цифр. Если количество цифр в аргументе меньше значения *precision*, выходное значение дополняется слева нулями. Значение	1

	не усечено, если число цифр превышает *точность*.	
S, s	Точность определяет максимальное количество выводимых символов. Символы, выходящие за рамки *precision*, не выводятся.	Длина строки (для asciiz – до нулевого символа)
%	-	-

3.2. fstr *

Этот тип данных представляет собой синоним для char * и является указателем на обработанную формат-строку. Значение, хранимое по указателю, не является простым массивом символов, а может так же содержать обработанные спецификации преобразования. Спецификация обрабатывается с символа %. Формат хранения обработанной спецификации представлен в таблице ниже (байты после 2 являются опциональными, их необходимость определяется по значениям флагов во 2 байте). Последовательность байт завершается нулевым байтом (без учета спецификации). В C++ – аналогичный класс FASM_Fstr. В ванильном FASM и FASM_OOP существует макрос fstr, который генерирует обработанную строку в компайл-тайме.

Таблица 4. Формат хранения обработанной спецификации

Номер байта	Номера бит	Значение
1	0-7	Символ %
2	0-3	Порядковый номер типа от 0, порядок нумерации соответствует таблице типов, кроме символа %, который имеет значение 15.
	4	Были ли указаны флаги
	5	Была ли указана ширина
	6	Была ли указана точность
	7	зарезервировано
3	0	Был ли указан флаг «-»
	1	Был ли указан флаг «+»
	2	Был ли указан флаг «0»
	3	Был ли указан флаг « »
	4-7	Зарезервировано
4-7	0-31	Значение ширины
8-11	0-31	Значение точности

3.3. fstr_dynamic

Сигнатура

fstr * fstr_dynamic(const char format[]);

Аргументы:

1. const char format[]

   Указатель на исходную строку формата.

Возвращаемое значение: массив байт формата.

Обрабатывает в рантайме исходную строку формата и выделяет под нее память.

В FASM доступен макрос fstr, создающий результирующую строку в компайл-тайме, но эту функцию вызвать для создания строки тоже возможно. Аналог в C++ - конструктор класса FASM_Fstr.

3.4. fstr_free

Сигнатура

void fstr_free(fstr * format);

Аргументы:

1. fstr * format

   Указатель на выделенный с помощью fstr_dynamic массив байт формата.

Возвращаемое значение: нет.

Освобождает ранее выделенную строку.

Аналог в C++ - деструктор класса FASM_Fstr.

4. Сборка

Сборка для asm проводилась на последней версии FLAT Assembler. Сборка для C и C++ проводилась на компиляторе gcc.

4.1. Сборка на C

Для сборки под C я использовал следующую строку:

gcc test_String:c -L. -l:String_dll_64.dll -o test_String:exe -nodefaultlibs -nostdlib

4.2. Сборка на C++

Для сборки под C++ я использовал следующую строку:

g++ test_String:cpp -L. -l:String_dll_64.dll -o test_String:exe -nostartfiles

4.3. Сборка под FASM

Сборка под ванильный фасм не требует каких-либо особых условий. Сборка под FASM_OOP потребует файлы в папке TOOLS в корне проекта, которые можно получить из репозитория https://github.com/ZReticules/FASM_OOP/tree/multiarch, ветки multiarch.