

Описание проекта `elements`

1. Цель проекта `elements` – предоставить набор функций для оперирования с битами, байтами, машинными словами.

2. Представление байт и слов в проекте

В составе стандартной библиотеки, начиная со стандарта C99, определены целочисленные типы с заданным количеством двоичных разрядов (к примеру, `uint8_t`). Однако проект `Elements` их не использует. Для определения байта использован `unsigned char`, для слова - `unsigned int`. Вследствие этого, требуются тесты для подтверждения кроссплатформенности программы.

3. Проекта состоит из двух файлов:

`elements.h` – заголовок, где объявлены константы и функции библиотеки.

`elements.c` – содержит определения функций, объявленных в `elements.h`.

4. Заголовок `elements.h`

4.1. Состав заголовка

Пользовательские типы (слово, байт, бит).

Константы (наибольшие значения, размеры, порядок хранения и представления, системы счисления при преобразовании байта в символы, параметры преобразования массива байт в символы).

Функции записи и чтения элементов.

Функции разбиения слова на массив байт, формирования слова из массива байт.

Функции преобразования байт и массива байт в массивы символов (без использования функции `spintf()` из стандартной библиотеки).

4.2. Пользовательские типы

`word` – слово (содержит натуральное число байт)

`byte` – байт (он может состоять и не из восьми бит)

`bit` – бит (один двоичный разряд)

4.3. Константы

Наибольшие значения:

| | |
|------------------|--|
| <i>WORD_MAX</i> | наибольшее значение слова |
| <i>BYTE_MAX</i> | наибольшее значение байта |
| <i>BIT_MAX</i> | наибольшее значение бита |
| <i>TETRA_MAX</i> | наибольшее значение тетрады (она состоит из четырёх двоичных разрядов) |

Размеры:

| | |
|----------------------------|---|
| <i>TETRA_SIZE_IN_BITS</i> | размер тетрады в битах (четыре) |
| <i>BYTE_SIZE_IN_BITS</i> | размер байта в битах |
| <i>WORD_SIZE_IN_BITS</i> | размер слова в битах |
| <i>BYTE_SIZE_IN_TETRAS</i> | размер байта в тетрадах (неполная тетрада тоже учитывается) |
| <i>WORD_SIZE_IN_BYTES</i> | размер слова в байтах (натуральное число) |

Порядок хранения и представления:

| | |
|------------------------|--|
| <i>endian_types</i> | тип для порядка представления: |
| <i>LITTLE_ENDIAN</i> | от младших разрядов к старшим |
| <i>BIG_ENDIAN</i> | от старших разрядов к младшим |
| <i>WORD_ENDIAN</i> | тип представления, принятый по умолчанию |
| <i>ENDIAN_NAMES[]</i> | массив, который содержит имена порядков хранения |

Значения бита:

| | |
|------------------|-------------|
| <i>BIT_SET</i> | бит равен 1 |
| <i>BIT_CLEAR</i> | бит равен 0 |

Доступные системы счисления при преобразовании байта в символы:

| | |
|--------------------|---------------------------------------|
| <i>number_base</i> | тип для представления байта: |
| <i>BASE_HEX</i> | в шестнадцатеричной системе счисления |
| <i>BASE_BIN</i> | в двоичной системе счисления |

Задание параметров преобразования массива байт в символы:

| | |
|-------------------|--|
| <i>trans_mode</i> | тег структуры для параметров преобразования, в её составе: |
|-------------------|--|

| | |
|---------------------------------|--|
| <i>number_base base</i> | система счисления: шестнадцатеричная, двоичная |
| <i>endian_types seq_endian</i> | порядок преобразования массива байтов |
| <i>endian_types byte_endian</i> | порядок преобразования отдельных байтов |
| <i>size_t gap</i> | число байт, после символов которых ставится символ-разделитель |
| <i>char gap_delim</i> | символ-разделитель группы байт (допускается любой, '\0' в том числе) |

4.4. Функции записи и чтения элементов

Запись и чтение элементов осуществляется в логическом порядке (7...0), а не в порядке представления. При оперировании с тетрадами значащими являются младшие четыре разряда.

word set_bitw(word w, byte bit_number, bit bit_value)

возвращает слово *w* с установленным значением бита под номером *bit_number* равным *bit_value*.

bit get_bitw(word w, byte bit_number)

возвращает значение бита под номером *bit_number* в слове *w*.

byte set_bitb(byte b, byte bit_number, bit bit_value)

возвращает байт *b* с установленным значением бита под номером *bit_number* равным *bit_value*.

bit get_bitb(byte b, byte bit_number)

возвращает значение бита под номером *bit_number* в байте *b*.

byte set_tetrab(byte b, size_t tetra_number, byte tetra_value)

возвращает байт *b* с установленным значением тетрады под номером *tetra_number* равным *tetra_value*.

byte get_tetrab(byte b, size_t tetra_number)

возвращает значение тетрады под номером *tetra_number* в байте *b*.

word set_bytew(word w, size_t byte_number, byte byte_value)

возвращает слово *w* с установленным значением байта под номером *byte_number* равным *byte_value*.

byte get_bytew(word w, size_t byte_number)

возвращает значение байта под номером *byte_number* в слове *w*.

4.5. Функции разбиения слова на массив байт, формирования слова из массива байт

*int split_word(byte *byte_mas, word srcw, endian_types dest_endian_type)*

Разбивает слово *srcw* на отдельные байты в логическом порядке и размещает их в массиве *byte_mas* в порядке *dest_endian_type*. Если порядок *LITTLE_ENDIAN* (*==0*), то в нулевом элементе располагается младший байт и далее по возрастанию. Если порядок *BIG_ENDIAN* (*!=0*), то в нулевом элементе располагается старший байт и далее по убыванию. Если возвращаемое значение больше или равно нулю, то это есть число преобразованных символов, иначе ошибка.

*word form_word(byte *byte_mas, endian_types src_endian_type)*

Создает слово из массива байт *byte_mas*. Слово формируется в порядке *src_endian_type*. Если порядок *LITTLE_ENDIAN* (*==0*), то в нулевом элементе располагается младший байт слова и далее по возрастанию. Если порядок *BIG_ENDIAN* (*!=0*), то в нулевом элементе располагается старший байт слова и далее по убыванию.

4.6. Функции преобразования байт и массива байт в массивы символов

Функция

char tetra_hex(byte tetra)

Преобразует младшую тетраду байта в символ шестнадцатеричного числа.

*int byte_hex(const byte b, char *s, const endian_types endian_type)*

Преобразует байт *b* в последовательность шестнадцатеричных цифр и записывает символы в строку *s*. Одна тетрада - одна цифра. Параметр *endian_type* определяет порядок вывода: *LITTLE_ENDIAN* (*==0*) младшая тетрада идет первой, *BIG_ENDIAN* (*!=0*) младшая тетрада идет последней. Возврат функции: *>= 0* - число записанных символов, *< 0* - ошибка.

*int byte_bin(const byte b, char *s, const endian_types endian_type)*

Преобразует байт *b* в последовательность двоичных цифр и записывает символы в строку *s*. Одна тетрада - четыре цифры. Параметр *endian_type* определяет порядок вывода: *LITTLE_ENDIAN* (*==0*) младший разряд идет первым, *BIG_ENDIAN* (*!=0*) младшая разряд идет последним. Возврат функции: *>= 0* - число символов, *< 0* - ошибка.

*int byte_base(const byte b, char *s, const endian_types endian_type, const number_base base)*

Преобразует байт *b* в последовательность двоичных или шестнадцатеричных цифр и записывает символы в строку *s*. Если возвращаемое значение больше или равно нулю, то это есть число преобразованных символов, иначе ошибка.

*int byte_trans(const byte *bm, char *s, size_t count, const struct trans_mode tm)*

Преобразует массив байт *bm* размером *count* в последовательность цифр и записывает символы в строку *s*. Формат преобразования определяется структурой *tm*. Если возвращаемое значение больше или равно нулю, то это есть число преобразованных символов, иначе ошибка.

*int word_trans (const word w, char *s, const struct trans_mode tm)*

Преобразует слово *w* в последовательность цифр и записывает символы в строку *s*. Формат преобразования определяется структурой *tm*. Если возвращаемое значение больше или равно нулю, то это есть число преобразованных символов, иначе ошибка.

*int word_hex(const word w, char *s)*

Преобразует слово *w* в последовательность шестнадцатеричных цифр и записывает символы в строку *s*. Порядок преобразования – прямой (*BIG_ENDIAN*). Если возвращаемое значение больше или равно нулю, то это есть число преобразованных символов, иначе ошибка.