Класс file

Редакция от 27.04.2026 · docs.kb-agava.ru

Содержание

1 Свойства 2
2 Методы 2
3 Методы записи 6
4 Пример использования 8

Класс предоставляет интерфейс для работы с файлами: открытие, закрытие, чтение и запись данных различных типов, позиционирование внутри файла. Поддерживает как текстовый, так и бинарный режимы работы.

Класс доступен начиная с версии `AS 1.4.5`.

1 Свойства

```
bool mostSignificantByteFirst
```

Назначение: Определяет порядок байтов (endianness) при чтении и записи числовых значений.

Аргументы: Отсутствуют.

Возвращаемое значение: Отсутствует (свойство).

Примечания:

- `true` – используется порядок **Big-endian** (старший байт первый).
- `false` (по умолчанию) – используется порядок **Little-endian** (младший байт первый).
- Влияет на методы `readInt`, `readUInt`, `readFloat`, `readDouble` и соответствующие методы записи.
- Может изменяться в любой момент до выполнения операций чтения/записи.

2 Методы

```
int open(const string &in filename, const string &in mode)
```

Назначение: Открывает файл для чтения и/или записи.

Аргументы:

- `filename` – путь к файлу (относительный или абсолютный).
- `mode` – режим открытия файла:
  - `"r"` – открыть для чтения (только чтение).
  - `"w"` – открыть для записи (создаёт новый или перезаписывает существующий).
  - `"a"` – открыть для добавления (запись в конец файла).

Возвращаемое значение:

- `0` – файл успешно открыт.
- `-1` – ошибка открытия (файл не существует, недостаточно прав, неверный режим).

Примечания:

- Перед открытием нового файла автоматически закрывает ранее открытый.
- На Windows автоматически добавляет флаг `"b"` (бинарный режим) для предотвращения преобразования `\r\n` в `\n`.
- На Windows CE пути преобразуются в абсолютные относительно директории приложения.
- Если режим не поддерживается (например, запись при `AS_WRITE_OPS == 0`), возвращает `-1`.

```
int close()
```

Назначение: Закрывает открытый файл.

Аргументы: Отсутствуют.

Возвращаемое значение:

- `0` – файл успешно закрыт.
- `-1` – файл уже был закрыт или не открыт.

Примечания:

- Автоматически вызывается в деструкторе объекта.
- После закрытия файла дальнейшие операции чтения/записи будут возвращать ошибки.

```
int getSize() const
```

Назначение: Возвращает размер файла в байтах.

Аргументы: Отсутствуют.

Возвращаемое значение:

- Размер файла в байтах (тип `int`).
- `-1` – файл не открыт или произошла ошибка.

Примечания:

- Для определения размера сохраняет текущую позицию, перемещается в конец файла и возвращается обратно.

```
bool isEndOfFile() const
```

Назначение: Проверяет, достигнут ли конец файла.

Аргументы: Отсутствуют.

Возвращаемое значение:

- `true` – достигнут конец файла или файл не открыт.
- `false` – не достигнут конец файла.

Примечания:

- Использует стандартную функцию `feof()`.
- Возвращает `true`, если файл не открыт.

```
int getPos() const
```

Назначение: Возвращает текущую позицию в файле.

Аргументы: Отсутствуют.

Возвращаемое значение:

- Текущая позиция в байтах от начала файла.

- `-1` – файл не открыт или произошла ошибка.

Примечания:

- Использует функцию `ftell()`.

```
int setPos(int pos)
```

Назначение: Устанавливает текущую позицию в файле от начала.

Аргументы:

- `pos` – абсолютная позиция в байтах от начала файла.

Возвращаемое значение:

- `0` – позиция успешно установлена.
- `-1` – файл не открыт или позиция вне допустимого диапазона.

Примечания:

- Использует функцию `fseek(SEEK_SET)`.

```
int movePos(int delta)
```

Назначение: Смещает текущую позицию в файле относительно текущего положения.

Аргументы:

- `delta` – смещение в байтах (положительное – вперёд, отрицательное – назад).

Возвращаемое значение:

- `0` – позиция успешно изменена.
- `-1` – файл не открыт или смещение выходит за границы.

Примечания:

- Использует функцию `fseek(SEEK_CUR)`.

```
string readString(uint length)
```

Назначение: Читает указанное количество символов из файла.

Аргументы:

- `length` – количество символов (байт) для чтения.

Возвращаемое значение:

- Строка с прочитанными данными (реальная длина может быть меньше `length`, если достигнут конец файла).
- Пустая строка – если файл не открыт.

Примечания:

- Возвращает все прочитанные байты как есть (включая нулевые символы).
- Размер результирующей строки соответствует количеству реально прочитанных байт.

```
string readLine()
```

Назначение: Читает одну строку из файла.

Аргументы: Отсутствуют.

Возвращаемое значение:

- Строка, содержащая прочитанную строку (без символа `\n`).
- Пустая строка – если достигнут конец файла или файл не открыт.

Примечания:

- Читает до символа перевода строки (`\n`) или до заполнения внутреннего буфера (255 символов).
- Поддерживает чтение длинных строк – операция повторяется до обнаружения `\n` или конца файла.
- Символ `\n` не включается в результат.

```
int64 readInt(uint bytes)
```

Назначение: Читает знаковое целое число из файла.

Аргументы:

- `bytes` – количество байт для чтения (1–8). Если `bytes > 8`, используется 8.

Возвращаемое значение:

- Прочитанное целое число (тип `int64`).
- `0` – если файл не открыт или достигнут конец файла.

Примечания:

- Поддерживает автоматическое расширение знака (sign extension) для отрицательных чисел.
- Порядок байт определяется свойством `mostSignificantByteFirst`.
- Младшие байты заполняются нулями, если `bytes < 8`.
- Для отрицательных чисел старшие байты заполняются `0xFF`.

```
uint64 readUInt(uint bytes)
```

Назначение: Читает беззнаковое целое число из файла.

Аргументы:

- `bytes` – количество байт для чтения (1–8). Если `bytes > 8`, используется 8.

Возвращаемое значение:

- Прочитанное беззнаковое целое число (тип `uint64`).
- `0` – если файл не открыт или достигнут конец файла.

Примечания:

- Порядок байт определяется свойством `mostSignificantByteFirst`.
- Не выполняет расширение знака.

- Младшие байты заполняются нулями, если `bytes < 8`.

```
float readFloat()
```

Назначение: Читает число с плавающей точкой (32 бита) из файла.

Аргументы: Отсутствуют.

Возвращаемое значение:

- Прочитанное число типа `float`.
- `0.0f` – если файл не открыт или достигнут конец файла.

Примечания:

- Читает ровно 4 байта.
- Порядок байт определяется свойством `mostSignificantByteFirst`.
- Использует объединение (union) для битового преобразования.

```
double readDouble()
```

Назначение: Читает число с плавающей точкой двойной точности (64 бита) из файла.

Аргументы: Отсутствуют.

Возвращаемое значение:

- Прочитанное число типа `double`.
- `0.0` – если файл не открыт или достигнут конец файла.

Примечания:

- Читает ровно 8 байт.
- Порядок байт определяется свойством `mostSignificantByteFirst`.
- Использует объединение (union) для битового преобразования.

3 Методы записи

```
int writeString(const string &in str)
```

Назначение: Записывает строку в файл.

Аргументы:

- `str` – строка для записи.

Возвращаемое значение:

- Количество записанных байт.
- `-1` – если файл не открыт.

Примечания:

- Записывает строку целиком, включая все символы (без завершающего нуля).

```
int writeInt(int64 val, uint bytes)
```

Назначение: Записывает знаковое целое число в файл.

Аргументы:

- `val` – значение для записи.
- `bytes` – количество байт для записи (1–8). Если `bytes > 8`, используется 8.

Возвращаемое значение:

- Количество записанных байт (0 или 1 в терминах `fwrite`).
- `0` – если файл не открыт.

Примечания:

- Порядок байт определяется свойством `mostSignificantByteFirst`.
- Записываются только младшие `bytes` байт числа.

```
int writeUInt(uint64 val, uint bytes)
```

Назначение: Записывает беззнаковое целое число в файл.

Аргументы:

- `val` – значение для записи.
- `bytes` – количество байт для записи (1–8). Если `bytes > 8`, используется 8.

Возвращаемое значение:

- Количество записанных байт (0 или 1 в терминах `fwrite`).
- `0` – если файл не открыт.

Примечания:

- Порядок байт определяется свойством `mostSignificantByteFirst`.
- Записываются только младшие `bytes` байт числа.

```
int writeFloat(float val)
```

Назначение: Записывает число с плавающей точкой (32 бита) в файл.

Аргументы:

- `val` – значение для записи.

Возвращаемое значение:

- Количество записанных байт (0 или 1 в терминах `fwrite`).
- `0` – если файл не открыт.

Примечания:

- Записывает ровно 4 байта.
- Порядок байт определяется свойством `mostSignificantByteFirst`.

```
int writeDouble(double val)
```

Назначение: Записывает число с плавающей точкой двойной точности (64 бита) в файл.

Аргументы:

- `val` – значение для записи.

Возвращаемое значение:

- Количество записанных байт (0 или 1 в терминах `fwrite`).
- `0` – если файл не открыт.

Примечания:

- Записывает ровно 8 байт.
- Порядок байт определяется свойством `mostSignificantByteFirst`.

4 Пример использования

```
// Создание и открытие файла для записи
file f;
f.mostSignificantByteFirst = false; // Little-endian

int result = f.open("data.bin", "w");

if (result == 0)
{
    f.writeString("Hello, World!");
    f.writeInt(12345, 4);
    f.writeFloat(3.14159f);
    f.close();
}

// Открытие файла для чтения
file f2;
f2.mostSignificantByteFirst = false;
f2.open("data.bin", "r");

if (!f2.isEndOfFile())
{
    string str = f2.readLine();
    int64 num = f2.readInt(4);
    float pi = f2.readFloat();
}

f2.close();

// Чтение бинарных данных с Big-endian порядком
file f3;
f3.mostSignificantByteFirst = true; // Network byte order
f3.open("network_data.bin", "r");
uint64 value = f3.readUInt(4); // Читает 4 байта как Big-endian
```