

# MPI SORT

Создано системой Doxygen 1.8.5

Пн 16 Дек 2013 21:48:18



# Оглавление

1	Список файлов	1
1.1	Файлы	1
2	Файлы	3
2.1	Файл arrayFree.c	3
2.1.1	Функции	3
2.1.1.1	arrayFree	3
2.2	Файл arrayInit.c	4
2.2.1	Функции	4
2.2.1.1	arrayInit	4
2.3	Файл core.c	6
2.3.1	Функции	6
2.3.1.1	core	6
2.4	Файл core.h	7
2.4.1	Макросы	8
2.4.1.1	MY_MPI_EVEN_N_EVEN	8
2.4.1.2	MY_MPI_QSORT	8
2.4.2	Функции	8
2.4.2.1	core	8
2.4.2.2	getAvg	9
2.4.2.3	globalIsSorted	10
2.4.2.4	isPowerOfTwo	11
2.4.2.5	MyEveSwapAndMerge	11
2.4.2.6	MyMpiEVESort	12
2.4.2.7	MyMpiQsort	12
2.4.2.8	MyQsortMpiSwap	13
2.4.2.9	MyQsortSwapAndMerge	14
2.5	Файл datatypes.h	15
2.5.1	Структуры данных	17
2.5.1.1	struct Border	17
2.5.1.2	struct Array	17

2.5.1.3	struct Ninja	18
2.5.1.4	struct Report	19
2.5.2	Макросы	19
2.5.2.1	ARRAY_INIT	19
2.5.2.2	DEFINED_RANDOM_MODE	19
2.5.2.3	fMax	19
2.5.2.4	fMin	19
2.5.2.5	NINJA_INIT	19
2.5.2.6	RANDOM_MODE	20
2.5.2.7	REPORT_INIT	20
2.5.3	Типы	20
2.5.3.1	array	20
2.5.3.2	border	20
2.5.3.3	ninja	20
2.5.3.4	report	20
2.5.4	Функции	20
2.5.4.1	arrayFree	20
2.5.4.2	arrayInit	20
2.5.4.3	fRand	21
2.5.4.4	getData	22
2.5.4.5	getNinja	22
2.5.4.6	getNinjaIdx	23
2.5.4.7	getNum	23
2.5.4.8	getSum	24
2.5.4.9	isSorted	25
2.5.4.10	MyBubbleSort	25
2.5.4.11	MyNormalizator	26
2.6	Файл getData.c	26
2.6.1	Функции	27
2.6.1.1	getData	27
2.7	Файл getNinja.c	28
2.7.1	Функции	28
2.7.1.1	getNinja	28
2.8	Файл getNinjaIdx.c	29
2.8.1	Функции	30
2.8.1.1	getNinjaIdx	30
2.9	Файл getNum.c	30
2.9.1	Функции	30
2.9.1.1	getNum	31
2.10	Файл globalIsSorted.c	32

2.10.1	Функции	33
2.10.1.1	globalIsSorted	33
2.11	Файл main.c	34
2.11.1	Функции	34
2.11.1.1	main	34
2.11.1.2	printGlobalReport	35
2.11.1.3	printReport	36
2.11.2	Переменные	37
2.11.2.1	myerror	37
2.12	Файл MyBubbleSort.c	38
2.12.1	Функции	38
2.12.1.1	MyBubbleSort	38
2.13	Файл myerrors.c	39
2.13.1	Макросы	40
2.13.1.1	LOCAL_SORT_UNsuc_STR	40
2.13.1.2	SORT_Suc_STR	40
2.13.1.3	SORT_UNsuc_STR	40
2.13.1.4	UNABLE_TO_ALLOC_MEM_STR	40
2.13.1.5	UNKNOWN_ERROR	40
2.13.1.6	UNKNOWN_MODE_STR	40
2.13.1.7	WRONG_PROC_NUM_STR	40
2.13.2	Функции	40
2.13.2.1	errorString	40
2.14	Файл myerrors.h	40
2.14.1	Макросы	41
2.14.1.1	LOCAL_SORT_UNsUccEsSEd	41
2.14.1.2	SORT_sUccEsSEd	41
2.14.1.3	SORT_UNsUccEsSEd	41
2.14.1.4	UNABLE_TO_ALLOCATE_MEMORY	41
2.14.1.5	UNKNOWN_MODE	41
2.14.1.6	WRONG_PROC_NUMBER	41
2.14.2	Функции	41
2.14.2.1	errorString	41
2.14.3	Переменные	42
2.14.3.1	myerror	42
2.15	Файл MyEveMpiSwapAndMerge.c	42
2.15.1	Функции	42
2.15.1.1	MyEveSwapAndMerge	43
2.16	Файл MyMpiEVESort.c	44
2.16.1	Функции	45

---

2.16.1.1	MyMpiEVESort	45
2.17	Файл MyMpiQsort.c	46
2.17.1	Функции	46
2.17.1.1	MyMpiQsort	46
2.18	Файл MyNormalizator.c	48
2.18.1	Функции	48
2.18.1.1	MyNormalizator	48
2.19	Файл MyQsortMpiSwap.c	49
2.19.1	Функции	49
2.19.1.1	MyQsortMpiSwap	49
2.20	Файл MyQsortSwapAndMerge.c	50
2.20.1	Функции	50
2.20.1.1	MyQsortSwapAndMerge	51
Алфавитный указатель		53

# Глава 1

## Список файлов

### 1.1 Файлы

Полный список файлов.

arrayFree.c	3
arrayInit.c	4
core.c	6
core.h	7
datatypes.h	15
getData.c	26
getNinja.c	28
getNinjaIdx.c	29
getNum.c	30
globalIsSorted.c	32
main.c	34
MyBubbleSort.c	38
myerrors.c	39
myerrors.h	40
MyEveMpiSwapAndMerge.c	42
MyMpiEVESort.c	44
MyMpiQsort.c	46
MyNormalizator.c	48
MyQsortMpiSwap.c	49
MyQsortSwapAndMerge.c	50





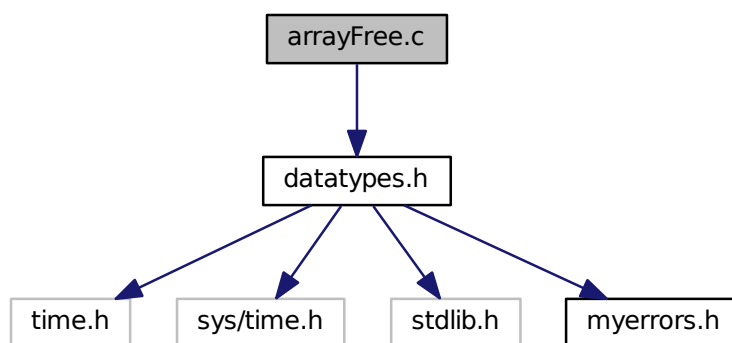
## Глава 2

# Файлы

### 2.1 Файл arrayFree.c

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для arrayFree.c:



#### Функции

- void `arrayFree` (`array *fullArray`)  
Освобождает память выделенную под массив

##### 2.1.1 Функции

2.1.1.1 void `arrayFree` ( `array * fullArray` )

Освобождает память выделенную под массив

Аргументы

fullArray	указатель на массив
-----------	---------------------

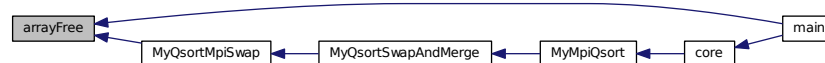
Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)

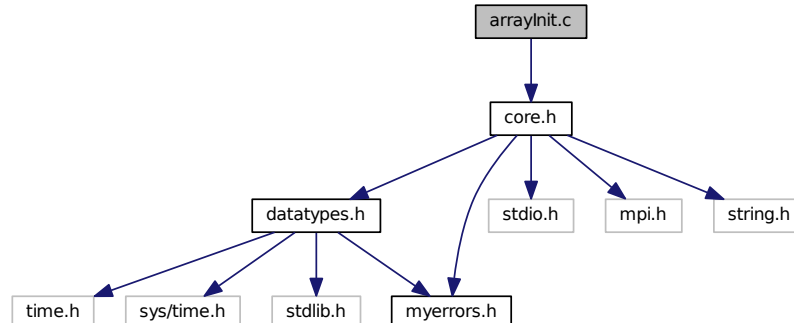
Граф вызова функции:



## 2.2 Файл arrayInit.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для arrayInit.c:



Функции

- void [arrayInit](#) ([array](#) \*emptyArray, int rank, int len, int ProcNum, int \*source, int mode)  
Инициализирует массив

### 2.2.1 Функции

2.2.1.1 void arrayInit ( array \* emptyArray, int rank, int len, int ProcNum, int \* source, int mode )

Инициализирует массив

## Аргументы

emptyArray	указатель на пустой массив
rank	номер процесса
len	длина последовательности
ProcNum	общее число процессов
source	указатель на источник данных
mode	режим работы

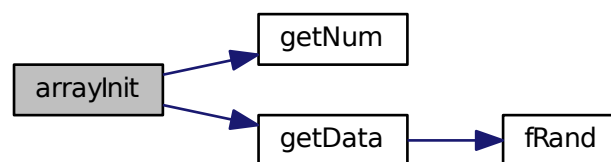
## Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)  
[border](#)

## Граф вызовов:



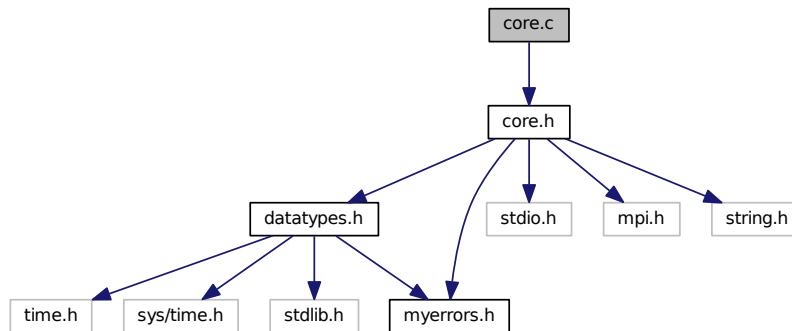
## Граф вызова функции:



## 2.3 Файл core.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для core.c:



### Функции

- `report core (array *myArray, int mode)`

Вычислительное ядро.

### 2.3.1 Функции

#### 2.3.1.1 `report core ( array * myArray, int mode )`

Вычислительное ядро.

В зависимости от режима производит различные сортировки

Аргументы

<code>myArray</code>	указатель на массив
<code>mode</code>	режим работы

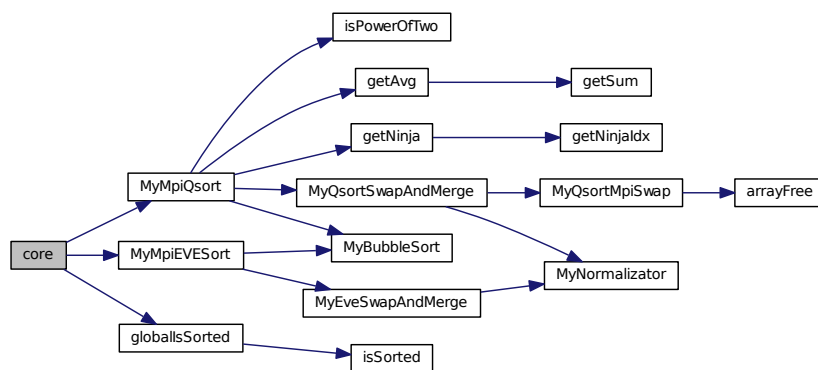
Автор

Arthur Asylgareev (Virid Raven)

Возвращает

время работы

Граф вызовов:



Граф вызова функции:



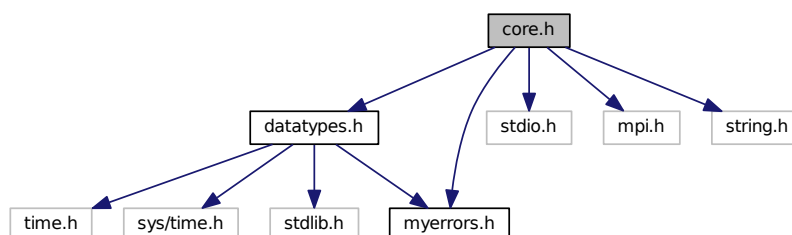
## 2.4 Файл core.h

```

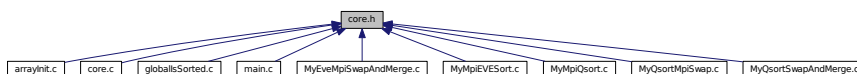
#include "datatypes.h"
#include "myerrors.h"
#include <stdio.h>
#include <mpi.h>
#include <string.h>

```

Граф включаемых заголовочных файлов для core.h:



Граф файлов, в которые включается этот файл:



## Макросы

- `#define MY_MPI_QSORT 1`
- `#define MY_MPI_EVEN_N_EVEN 2`

## Функции

- `int isPowerOfTwo (unsigned int x)`  
Проверка является ли число степенью двойки
- `double getAvg (array *myArray, MPI_Comm currentComm)`  
Ищет среднее арифметическое массива разбросанного по процессам
- `report core (array *myArray, int mode)`  
Вычислительное ядро.
- `void MyQsortSwapAndMerge (array *myArray, ninja *myNinja, int rank, int ProcNum, MPI_Comm currentComm)`  
Обмен и слияние массивов между процессами при "быстрой" сортировке
- `void MyEveSwapAndMerge (array *myArray, int rank, MPI_Comm currentComm)`  
Обмен и слияние массивов между процессами при чет-нечетной сортировке
- `void MyQsortMpiSwap (array *myArray, int ninjaIdx, int inlen, int outlen, int rank, int ProcNum, MPI_Comm currentComm)`  
Обмен элементами массивов между процессами при "быстрой" сортировке
- `int globalIsSorted (array *myArray, MPI_Comm currentComm)`  
Глобальная проверка на отсортированность массива разбросанного по процессам
- `int MyMpiQsort (array *myArray)`  
Организует параллельную "быструю" сортировку массива
- `int MyMpiEVESort (array *myArray)`  
Организует параллельную чет-нечетную сортировку массива

### 2.4.1 Макросы

2.4.1.1 `#define MY_MPI_EVEN_N_EVEN 2`

2.4.1.2 `#define MY_MPI_QSORT 1`

### 2.4.2 Функции

2.4.2.1 `report core ( array * myArray, int mode )`

Вычислительное ядро.

В зависимости от режима производит различные сортировки

Аргументы

myArray	указатель на массив
mode	режим работы

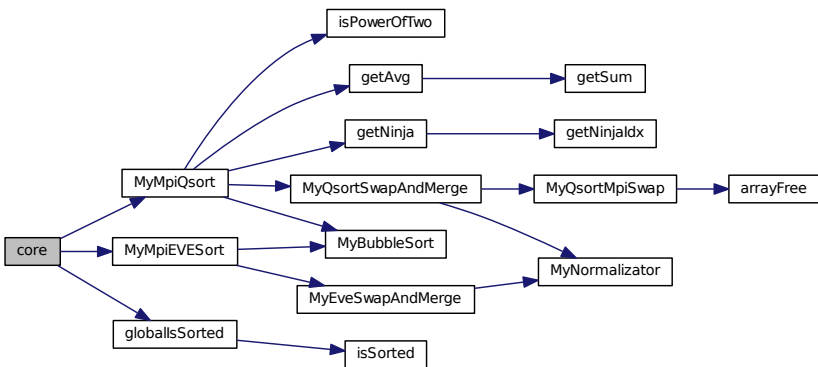
Автор

Arthur Asylgareev (Virid Raven)

Возвращает

время работы

Граф вызовов:



Граф вызова функции:



2.4.2.2 double getAvg ( array \* myArray, MPI\_Comm currentComm ) [inline]

Ищет среднее арифметическое массива разбросанного по процессам

Аргументы

myArray	указатель на массив
currentComm	коммуникатор

Автор

Arthur Asylgareev (Virid Raven)

Возвращает

среднее арифметическое

Граф вызовов:



Граф вызова функции:



2.4.2.3 `int globalIsSorted ( array * myArray, MPI_Comm currentComm )`

Глобальная проверка на отсортированность массива разбросанного по процессам

Аргументы

<code>myArray</code>	указатель на массив
<code>currentComm</code>	коммуникатор для которого проверяется упорядоченность

Автор

Arthur Asylgareev (Virid Raven)

Возвращает

результат сортировки

Граф вызовов:





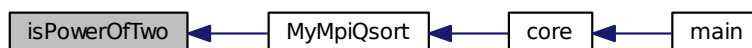
Граф вызова функции:



2.4.2.4 `int isPowerOfTwo ( unsigned int x ) [inline]`

Проверка является ли число степенью двойки

Граф вызова функции:



2.4.2.5 `void MyEveSwapAndMerge ( array * myArray, int rank, MPI_Comm currentComm )`

бмен и слияние массивов между процессами при чет-нечетной сортировке

Аргументы

myArray	указатель на массив
rank	номер процесса
currentComm	коммуникатор

Автор

Arthur Asylgareev (Virid Raven)

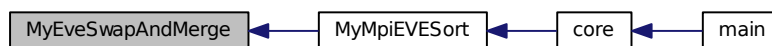
См. также

[MyMpiEVESort\(\)](#)

Граф вызовов:



Граф вызова функции:



#### 2.4.2.6 int MyMpiEVESort ( array \* myArray )

Организует параллельную чет-неченую сортировку массива

Аргументы

myArray	указатель на массив
---------	---------------------

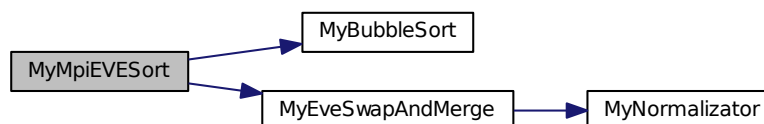
Автор

Arthur Asylgareev (Virid Raven)

Возвращает

число шагов

Граф вызовов:



Граф вызова функции:



#### 2.4.2.7 int MyMpiQsort ( array \* myArray )

Организует параллельную "быструю" сортировку массива

## Аргументы

myArray	указатель на массив
---------	---------------------

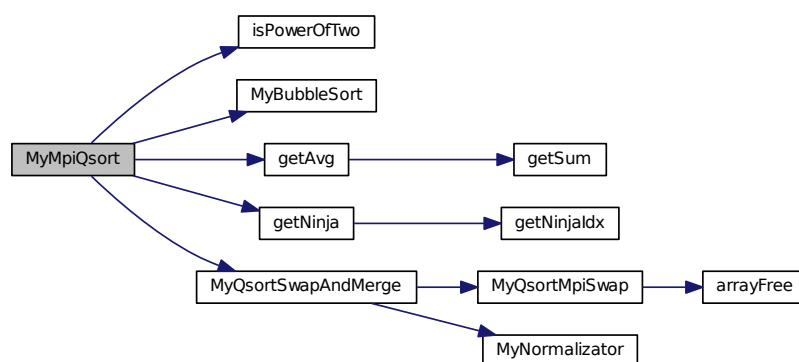
## Автор

Arthur Asylgareev (Virid Raven)

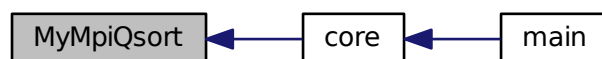
## Возвращает

число шагов

## Граф вызовов:



## Граф вызова функции:



2.4.2.8 void MyQsortMpiSwap ( array \* myArray, int ninjaIdx, int inlen, int outlen, int rank, int ProcNum, MPI\_Comm currentComm )

Обмен элементами массивов между процессами при "быстрой" сортировке

## Аргументы

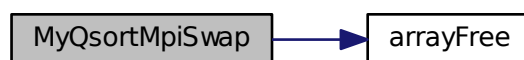
myArray	указатель на массив
ninjaIdx	индекс ведущего элемента

inlen	количество принимаемых элементов
outlen	количество отправляемых элементов
rank	номер процесса
ProcNum	общее число процессов
currentComm	коммуникатор

Автор

Arthur Asylgareev (Virid Raven)

Граф вызовов:



Граф вызова функции:



2.4.2.9 void MyQsortSwapAndMerge ( array \* myArray, ninja \* myNinja, int rank, int ProcNum, MPI\_Comm currentComm )

Обмен и слияние массивов между процессами при "быстрой" сортировке

Аргументы

myArray	указатель на массив
myNinja	указатель на ниндзю
rank	номер процесса
ProcNum	общее число процессов
currentComm	коммуникатор

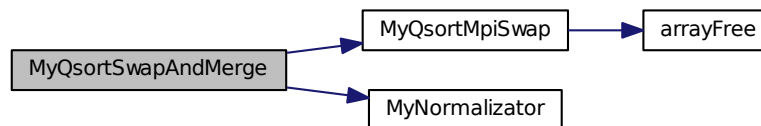
Автор

Arthur Asylgareev (Virid Raven)

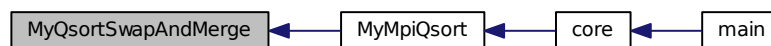
См. также

[MyMpiQsort\(\)](#)

Граф вызовов:



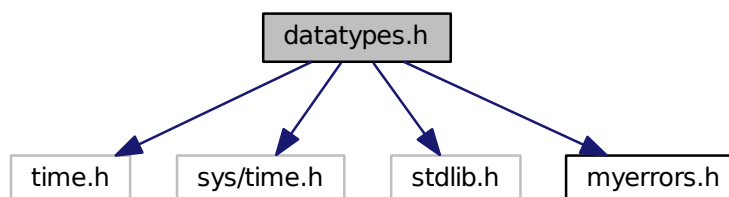
Граф вызова функции:



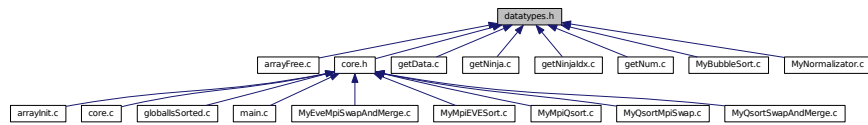
## 2.5 Файл datatypes.h

```
#include <time.h>
#include <sys/time.h>
#include <stdlib.h>
#include "myerrors.h"
```

Граф включаемых заголовочных файлов для datatypes.h:



Граф файлов, в которые включается этот файл:



## Структуры данных

- struct [Border](#)  
Граница разбиений [Подробнее...](#)
- struct [Array](#)  
Массив, знающий свою длину [Подробнее...](#)
- struct [Ninja](#)  
Структура хранящая индекс первого элемента, большего чем ведущий [Подробнее...](#)
- struct [Report](#)  
Структура-отчет [Подробнее...](#)

## Макросы

- `#define` [ARRAY\\_INIT](#) {0, NULL}
- `#define` [NINJA\\_INIT](#) {0, 0}
- `#define` [REPORT\\_INIT](#) {0, [UNKNOWN\\_MODE](#), 0, 0, 0, "Unknown mode"}
- `#define` [RANDOM\\_MODE](#) 3
- `#define` [DEFINED\\_RANDOM\\_MODE](#) 4
- `#define` [fMin](#) -1000.
- `#define` [fMax](#) 1000.

## Определения типов

- `typedef struct` [Border](#) [border](#)
- `typedef struct` [Array](#) [array](#)
- `typedef struct` [Ninja](#) [ninja](#)
- `typedef struct` [Report](#) [report](#)

## Функции

- double [fRand](#) ()  
Генерирует случайное число в границах от [fMin](#) до [fMax](#).
- int [isSorted](#) ([array](#) \*misticArray)  
Проверят отсортирован ли массив
- double [getSum](#) ([array](#) \*myArray)  
Сумма всех элементов
- [border](#) [getNum](#) (int rank, int N, int size)  
Получить границы разбиений
- double \* [getData](#) ([border](#) \*slice, void \*source, int mode)  
Возвращает массив с данными
- int [MyBubbleSort](#) ([array](#) \*unsortedArray)  
Пузырьковая сортировка

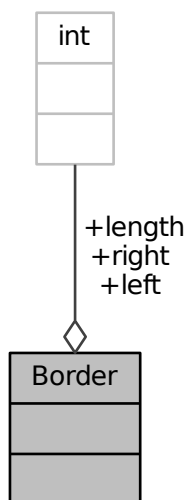
- `int MyNormalizator (array *firstArray)`  
Организует слияние массива, более эффективное по временным затратам чем пузырьковая сортировка
- `void arrayInit (array *emptyArray, int rank, int N, int size, int *seed, int mode)`  
Инициализирует массив
- `void arrayFree (array *emptyArray)`  
Освобождает память выделенную под массив
- `int getNinjaIdx (array *wholeArray, double lider)`  
Ищет индекс первого элемента большего чем ведущий
- `ninja getNinja (array *wholeArray, double lider)`  
Возвращает структуру хранящую индекс первого элемента большего, чем ведущий

## 2.5.1 Структуры данных

### 2.5.1.1 struct Border

Граница разбиений

Граф связей класса Border:



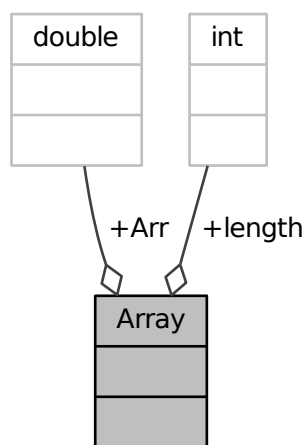
Поля структур

int	left	Левая граница разбиения
int	length	Длина разбиения
int	right	Правая граница разбиения

### 2.5.1.2 struct Array

Массив, знающий свою длину

Граф связей класса Array:



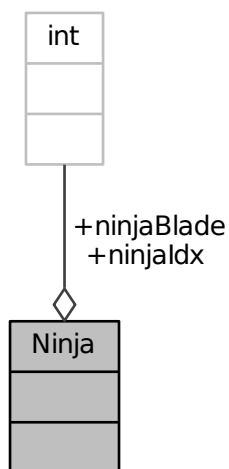
Поля структур

double *	Arr	Указатель на массив
int	length	Длина массива

#### 2.5.1.3 struct Ninja

Структура хранящая индекс первого элемента, большего чем ведущий

Граф связей класса Ninja:





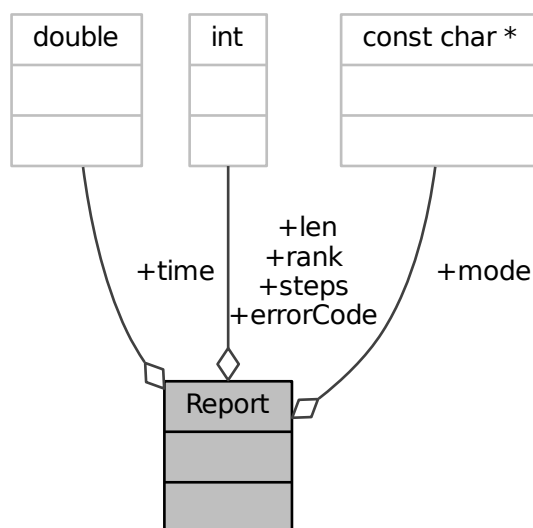
Поля структур

int	ninjaBlade	Длина отрезка от элемента до конца массива
int	ninjaIdx	Индекс элемента

#### 2.5.1.4 struct Report

Структура-отчет

Граф связей класса Report:



Поля структур

int	errorCode	Код ошибки
int	len	Длина массива на процессе
const char *	mode	Режим работы
int	rank	Номер процесса
int	steps	Число шагов алгоритма
double	time	Время работы

## 2.5.2 Макросы

2.5.2.1 `#define ARRAY_INIT {0, NULL}`

2.5.2.2 `#define DEFINED_RANDOM_MODE 4`

2.5.2.3 `#define fMax 1000.`

2.5.2.4 `#define fMin -1000.`

2.5.2.5 `#define NINJA_INIT {0, 0}`

2.5.2.6 `#define RANDOM_MODE 3`

2.5.2.7 `#define REPORT_INIT {0, UNKNOWN_MODE,0,0,0,"Unknown mode"}`

### 2.5.3 Типы

2.5.3.1 `typedef struct Array array`

2.5.3.2 `typedef struct Border border`

2.5.3.3 `typedef struct Ninja ninja`

2.5.3.4 `typedef struct Report report`

### 2.5.4 Функции

2.5.4.1 `void arrayFree ( array * fullArray )`

Освобождает память выделенную под массив

Аргументы

<code>fullArray</code>	указатель на массив
------------------------	---------------------

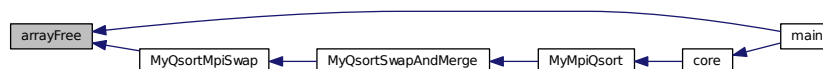
Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)

Граф вызова функции:



2.5.4.2 `void arrayInit ( array * emptyArray, int rank, int len, int ProcNum, int * source, int mode )`

Инициализирует массив

Аргументы

<code>emptyArray</code>	указатель на пустой массив
<code>rank</code>	номер процесса
<code>len</code>	длина последовательности
<code>ProcNum</code>	общее число процессов

source	указатель на источник данных
mode	режим работы

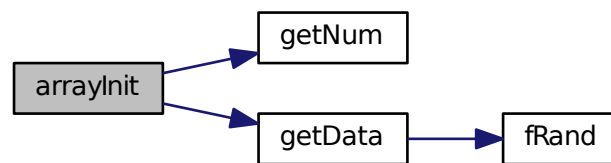
Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)  
[border](#)

Граф вызовов:



Граф вызова функции:



#### 2.5.4.3 `double fRand ( ) [inline]`

Генерирует случайное число в границах от `fMin` до `fMax`.

Возвращает

случайное число

Граф вызова функции:



2.5.4.4 `double* getData ( border * slice, void * source, int mode )`

Возвращает массив с данными

Аргументы

slice	границы
source	источник данных
mode	режим работы

Автор

Arthur Asylgareev (Virid Raven)

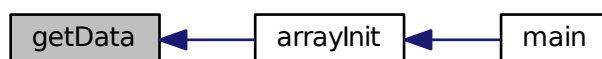
Возвращает

указатель на массив

Граф вызовов:



Граф вызова функции:



2.5.4.5 `ninja getNinja ( array * wholeArray, double lider )`

Возвращает структуру хранящую индекс первого элемента большего, чем ведущий

Аргументы

wholeArray	указатель на массив
lider	ведущий элемент

Возвращает

временный ниндзя

Автор

Arthur Asylgareev (Virid Raven)

См. также

[ninja](#)

Граф вызовов:



Граф вызова функции:



2.5.4.6 `int getNinjalDx ( array * wholeArray, double lider )`

Ищет индекс первого элемента большего чем ведущий

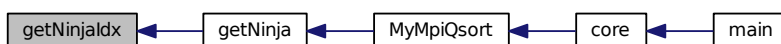
Аргументы

<code>wholeArray</code>	указатель на массив
<code>lider</code>	ведущий элемент

Возвращает

индекс первого элемента большего чем ведущий

Граф вызова функции:



2.5.4.7 `border getNum ( int rank, int N, int size )`

Получить границы разбиений

## Аргументы

rank	номер процесса
N	число элементов
size	число процессов

## Автор

Arthur Asylgareev (Virid Raven)

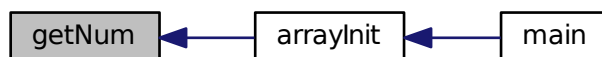
## Возвращает

границы разбиений

## См. также

[border](#)

## Граф вызова функции:



## 2.5.4.8 double getSum ( array \* myArray ) [inline]

## Сумма всех элементов

## Аргументы

misticArray	указатель на массив
-------------	---------------------

## Возвращает

сумма всех элементов массива

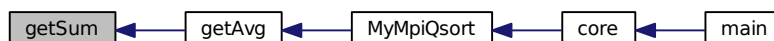
## Автор

Arthur Asylgareev (Virid Raven)

## См. также

[array](#)

## Граф вызова функции:



2.5.4.9 `int isSorted ( array * misticArray ) [inline]`

Проверят отсортирован ли массив

Аргументы

<code>misticArray</code>	указатель на массив
--------------------------	---------------------

Возвращает

результат проверки

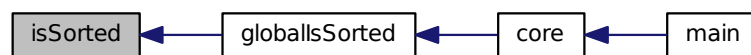
Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)

Граф вызова функции:

2.5.4.10 `int MyBubbleSort ( array * unsortedArray )`

Пузырьковая сортировка

Аргументы

<code>unsortedArray</code>	указатель на неотсортированный массив
----------------------------	---------------------------------------

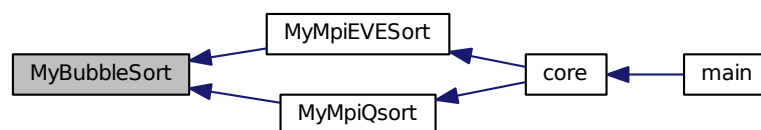
Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)

Граф вызова функции:



#### 2.5.4.11 int MyNormalizator ( array \* firstArray )

Организует слияние массива, более эффективное по временным затратам чем пузырьковая сортировка

Аргументы

firstArray	указатель на массив
------------	---------------------

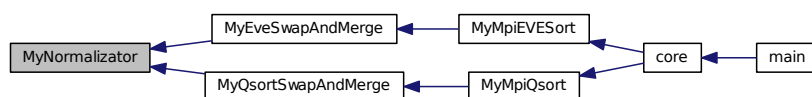
Автор

Arthur Asylgareev (Virid Raven)

Возвращает

число перестановок

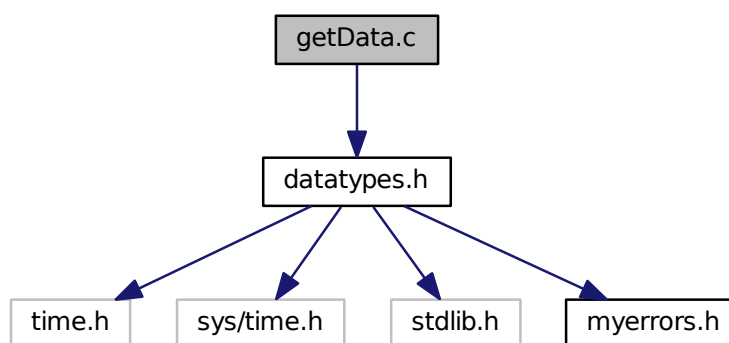
Граф вызова функции:



## 2.6 Файл getData.c

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для getData.c:



Функции

- double \* [getData](#) (border \*slice, void \*source, int mode)

Возвращает массив с данными



### 2.6.1 Функции

#### 2.6.1.1 double\* getData ( border \* slice, void \* source, int mode )

Возвращает массив с данными

Аргументы

slice	границы
source	источник данных
mode	режим работы

Автор

Arthur Asylgareev (Virid Raven)

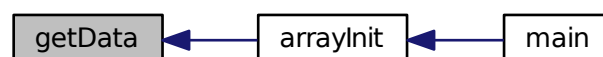
Возвращает

указатель на массив

Граф вызовов:



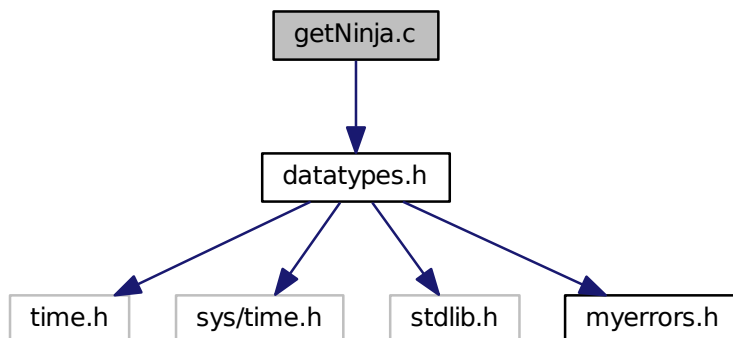
Граф вызова функции:



## 2.7 Файл getNinja.c

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для getNinja.c:



### Функции

- [ninja getNinja](#) ([array](#) \*wholeArray, double lider)

Возвращает структуру хранящую индекс первого элемента большего, чем ведущий

### 2.7.1 Функции

#### 2.7.1.1 [ninja getNinja](#) ( [array](#) \* wholeArray, double lider )

Возвращает структуру хранящую индекс первого элемента большего, чем ведущий

Аргументы

wholeArray	указатель на массив
lider	ведущий элемент

Возвращает

временный ниндзя

Автор

Arthur Asylgareev (Virid Raven)

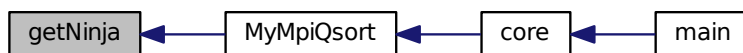
См. также

[ninja](#)

Граф вызовов:



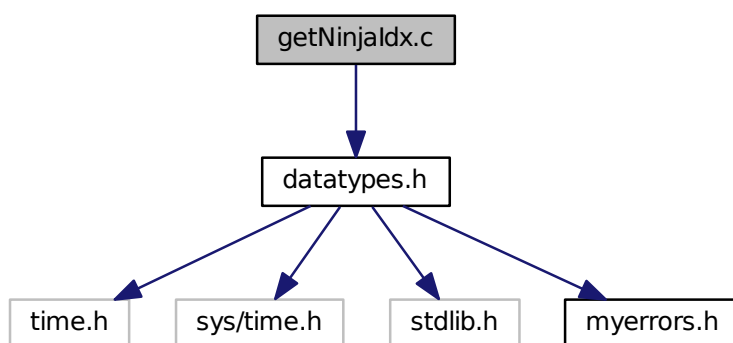
Граф вызова функции:



## 2.8 Файл getNinjaIdx.c

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для getNinjaIdx.c:



Функции

- `int getNinjaIdx (array *wholeArray, double lider)`  
Ищет индекс первого элемента большего чем ведущий

## 2.8.1 Функции

### 2.8.1.1 `int getNinjaIdx ( array * wholeArray, double lider )`

Ищет индекс первого элемента большего чем ведущий

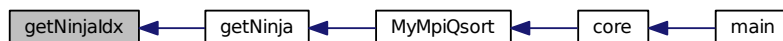
Аргументы

<code>wholeArray</code>	указатель на массив
<code>lider</code>	ведущий элемент

Возвращает

индекс первого элемента большего чем ведущий

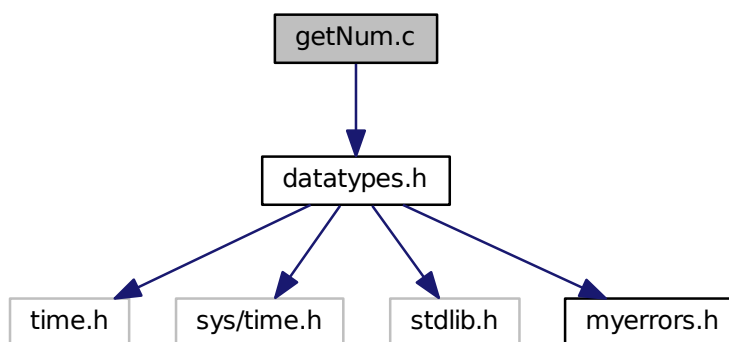
Граф вызова функции:



## 2.9 Файл `getNum.c`

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для `getNum.c`:



Функции

- `border getNum (int rank, int N, int size)`  
Получить границы разбиений

### 2.9.1 Функции

2.9.1.1 border getNum ( int rank, int N, int size )

Получить границы разбиений

## Аргументы

rank	номер процесса
N	число элементов
size	число процессов

## Автор

Arthur Asylgareev (Virid Raven)

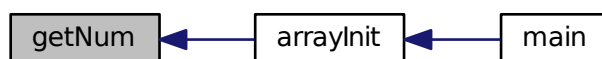
## Возвращает

границы разбиений

## См. также

[border](#)

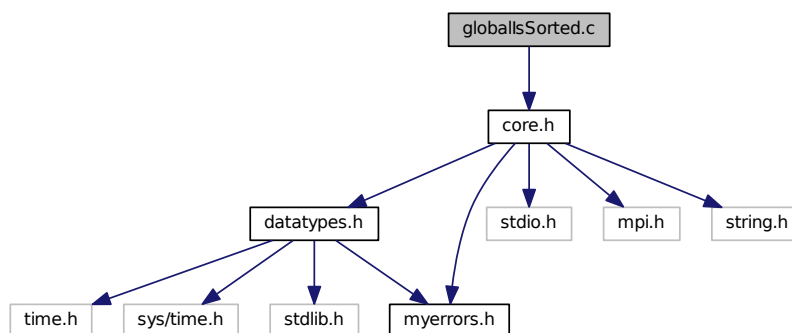
## Граф вызова функции:



## 2.10 Файл globalIsSorted.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для globalIsSorted.c:



## Функции

- `int globalIsSorted (array *myArray, MPI_Comm currentComm)`

Глобальная проверка на отсортированность массива разбросанного по процессам

### 2.10.1 Функции

#### 2.10.1.1 `int globalIsSorted ( array * myArray, MPI_Comm currentComm )`

Глобальная проверка на отсортированность массива разбросанного по процессам

Аргументы

<code>myArray</code>	указатель на массив
<code>currentComm</code>	коммуникатор для которого проверяется упорядоченность

Автор

Arthur Asylgareev (Virid Raven)

Возвращает

результат сортировки

Граф вызовов:



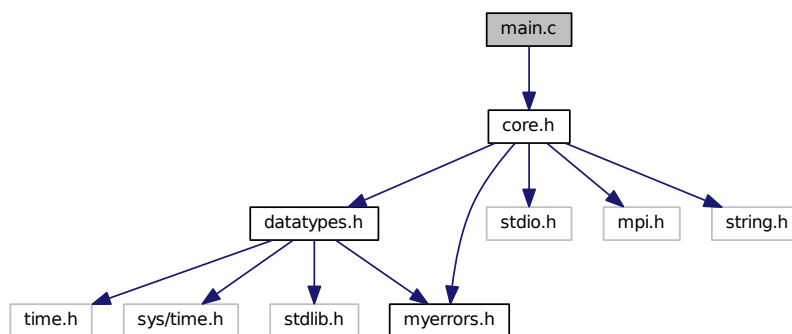
Граф вызова функции:



## 2.11 Файл main.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для main.c:



### Функции

- void `printReport` (`report *myReport`)

Выводит на экран отчет по работе процесса

- void `printGlobalReport` (`report *myReport`, int len)

Выводит на экран глобальный отчет по работе процесса

- int `main` (int argc, char \*argv[])

Точка входа

### Переменные

- int `myerror` =0

код ошибки

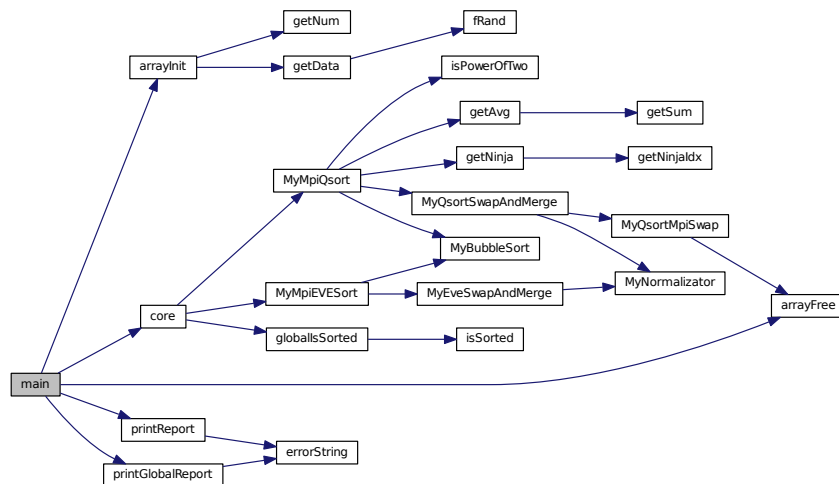
#### 2.11.1 Функции

##### 2.11.1.1 int main ( int argc, char \* argv[] )

Точка входа



Граф вызовов:



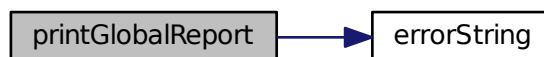
2.11.1.2 void printGlobalReport ( report \* myReport, int len ) [inline]

Выводит на экран глобальный отчет по работе процесса

Аргументы

myReport	указатель на отчет
len	длина последовательности элементов

Граф вызовов:



Граф вызова функции:



2.11.1.3 void printReport ( report \* myReport ) [inline]

Выводит на экран отчет по работе процесса

## Аргументы

myReport	указатель на отчет
----------	--------------------

## Автор

Arthur Asylgareev (Virid Raven)

## Граф вызовов:



## Граф вызова функции:



## 2.11.2 Переменные

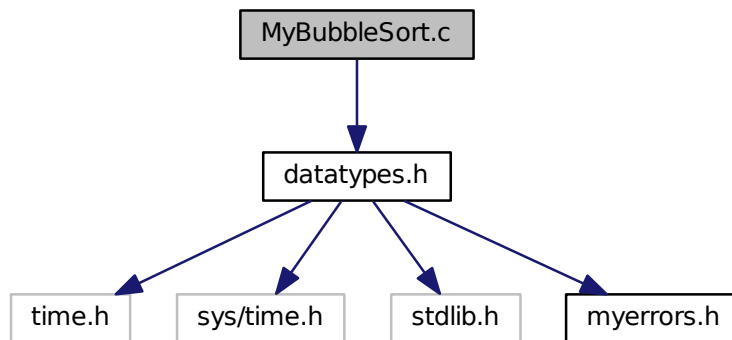
## 2.11.2.1 int myerror =0

код ошибки

## 2.12 Файл MyBubbleSort.c

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для MyBubbleSort.c:



### Функции

- `int MyBubbleSort (array *unsortedArray)`

Пузырьковая сортировка

### 2.12.1 Функции

2.12.1.1 `int MyBubbleSort ( array * unsortedArray )`

Пузырьковая сортировка

Аргументы

<code>unsortedArray</code>	указатель на неотсортированный массив
----------------------------	---------------------------------------

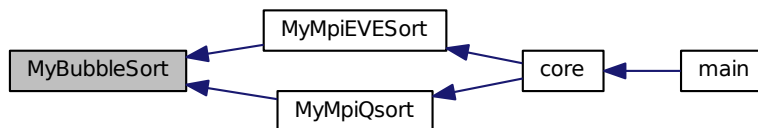
Автор

Arthur Asylgareev (Virid Raven)

См. также

[array](#)

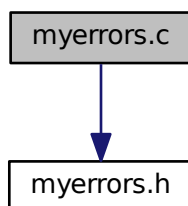
Граф вызова функции:



## 2.13 Файл myerrors.c

```
#include "myerrors.h"
```

Граф включаемых заголовочных файлов для myerrors.c:



### Макросы

- `#define SORT_SUC_STR` "Global sort succeeded!"
- `#define SORT_UNSUC_STR` "Global sort doesn't succeeded!"
- `#define LOCAL_SORT_UNSUC_STR` "Local sort doesn't succeeded!"
- `#define WRONG_PROC_NUM_STR` "Num of processes isn't power of 2!"
- `#define UNKNOWN_MODE_STR` "Work mode is unknown!"
- `#define UNABLE_TO_ALLOC_MEM_STR` "Unable to allocate memory!"
- `#define UNKNOWN_ERROR` "Unknown error occurred!"

### Функции

- `const char * errorString` (int errorcode)

Преобразует код ошибки в строковое представление

### 2.13.1 Макросы

2.13.1.1 `#define LOCAL_SORT_UNsuc_STR "Local sort doesn't succeeded!"`

2.13.1.2 `#define SORT_Suc_STR "Global sort succeeded!"`

2.13.1.3 `#define SORT_UNsuc_STR "Global sort doesn't succeeded!"`

2.13.1.4 `#define UNABLE_TO_ALLOC_MEM_STR "Unable to allocate memory!"`

2.13.1.5 `#define UNKNOWN_ERROR "Unknown error occurred!"`

2.13.1.6 `#define UNKNOWN_MODE_STR "Work mode is unknown!"`

2.13.1.7 `#define WRONG_PROC_NUM_STR "Num of processes isn't power of 2!"`

### 2.13.2 Функции

2.13.2.1 `const char* errorString ( int errorcode )`

Преобразует код ошибки в строковое представление

строковое представление ошибки

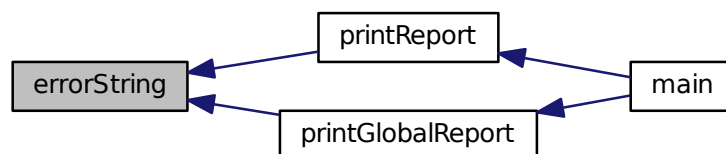
Аргументы

errorcode	код ошибки
-----------	------------

Возвращает

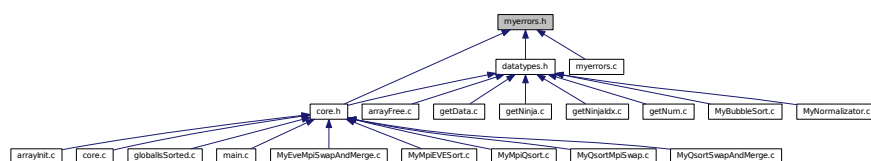
строковое представление ошибки

Граф вызова функции:



## 2.14 Файл myerrors.h

Граф файлов, в которые включается этот файл:



## Макросы

- `#define SORT_SUCCEEDED 0`
- `#define SORT_UNSUCCEEDED -1`
- `#define WRONG_PROC_NUMBER -2`
- `#define LOCAL_SORT_UNSUCCEEDED -3`
- `#define UNKNOWN_MODE -4`
- `#define UNABLE_TO_ALLOCATE_MEMORY -5`

## Функции

- `const char * errorString (int errorcode)`  
строковое представление ошибки

## Переменные

- `int myerror`  
код ошибки

### 2.14.1 Макросы

- 2.14.1.1 `#define LOCAL_SORT_UNSUCCEEDED -3`
- 2.14.1.2 `#define SORT_SUCCEEDED 0`
- 2.14.1.3 `#define SORT_UNSUCCEEDED -1`
- 2.14.1.4 `#define UNABLE_TO_ALLOCATE_MEMORY -5`
- 2.14.1.5 `#define UNKNOWN_MODE -4`
- 2.14.1.6 `#define WRONG_PROC_NUMBER -2`

### 2.14.2 Функции

- 2.14.2.1 `const char* errorString ( int errorcode )`

строковое представление ошибки

строковое представление ошибки

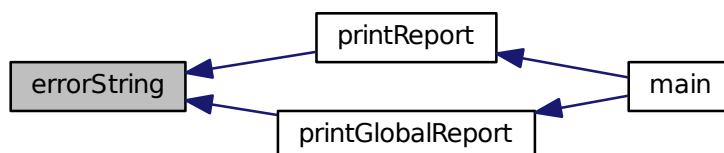
#### Аргументы

errorcode	код ошибки
-----------	------------

Возвращает

строковое представление ошибки

Граф вызова функции:



## 2.14.3 Переменные

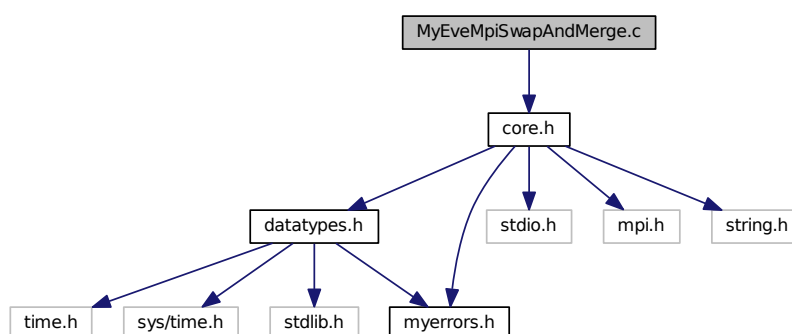
### 2.14.3.1 int myerror

код ошибки

## 2.15 Файл MyEveMpiSwapAndMerge.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для MyEveMpiSwapAndMerge.c:



Функции

- void [MyEveSwapAndMerge](#) ([array](#) \*myArray, int rank, MPI\_Comm currentComm)  
бмен и слияние массивов между процессами при чет-нечетной сортировке

### 2.15.1 Функции



2.15.1.1 void MyEveSwapAndMerge ( array \* myArray, int rank, MPI\_Comm currentComm )

бмен и слияние массивов между процессами при чет-нечетной сортировке

## Аргументы

myArray	указатель на массив
rank	номер процесса
currentComm	коммуникатор

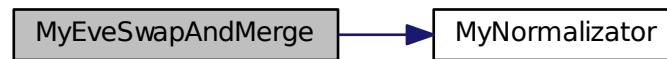
## Автор

Arthur Asylgareev (Virid Raven)

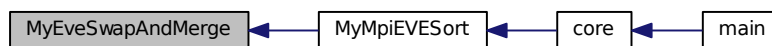
См. также

[MyMpiEVESort\(\)](#)

Граф вызовов:



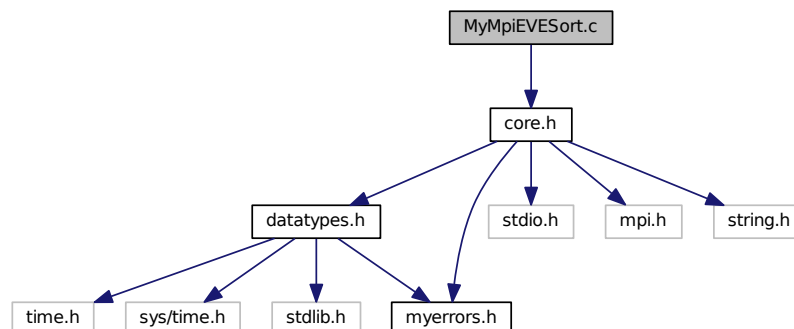
Граф вызова функции:



## 2.16 Файл MyMpiEVESort.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для MyMpiEVESort.c:



## Функции

- `int MyMpiEVESort (array *myArray)`

Организует параллельную чет-неченую сортировку массива

## 2.16.1 Функции

2.16.1.1 `int MyMpiEVESort ( array * myArray )`

Организует параллельную чет-неченую сортировку массива

## Аргументы

<code>myArray</code>	указатель на массив
----------------------	---------------------

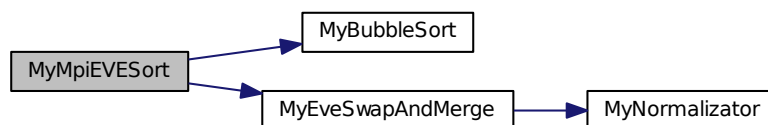
## Автор

Arthur Asylgareev (Virid Raven)

## Возвращает

число шагов

## Граф вызовов:



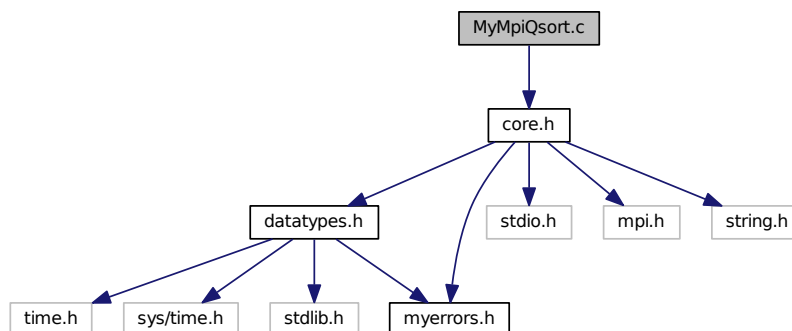
## Граф вызова функции:



## 2.17 Файл MyMpiQsort.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для MyMpiQsort.c:



### Функции

- `int MyMpiQsort (array *myArray)`

Организует параллельную "быструю" сортировку массива

### 2.17.1 Функции

#### 2.17.1.1 `int MyMpiQsort ( array * myArray )`

Организует параллельную "быструю" сортировку массива

Аргументы

<code>myArray</code>	указатель на массив
----------------------	---------------------

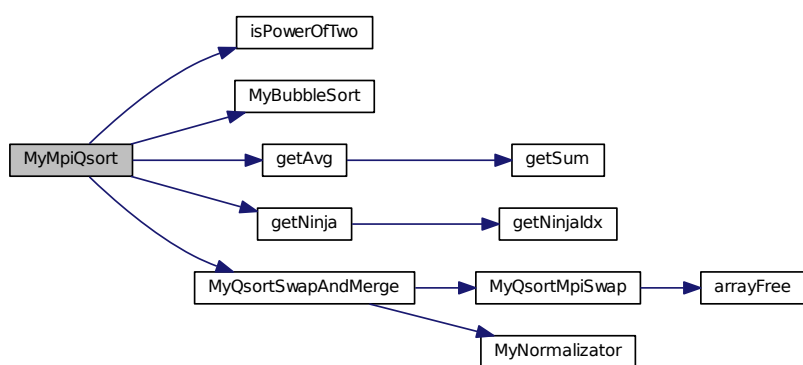
Автор

Arthur Asylgareev (Virid Raven)

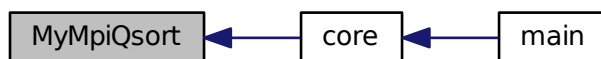
Возвращает

число шагов

Граф вызовов:



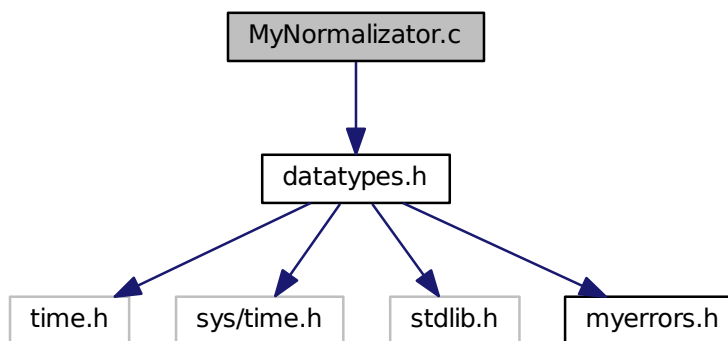
Граф вызова функции:



## 2.18 Файл MyNormalizator.c

```
#include "datatypes.h"
```

Граф включаемых заголовочных файлов для MyNormalizator.c:



### Функции

- int `MyNormalizator` (`array *firstArray`)

Организует слияние массива, более эффективное по временным затратам чем пузырьковая сортировка

### 2.18.1 Функции

#### 2.18.1.1 int MyNormalizator ( array \* firstArray )

Организует слияние массива, более эффективное по временным затратам чем пузырьковая сортировка

Аргументы

<code>firstArray</code>	указатель на массив
-------------------------	---------------------

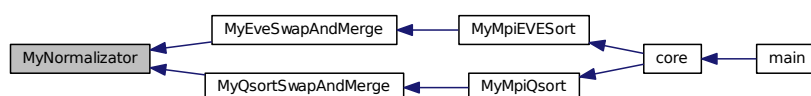
Автор

Arthur Asylgareev (Virid Raven)

Возвращает

число перестановок

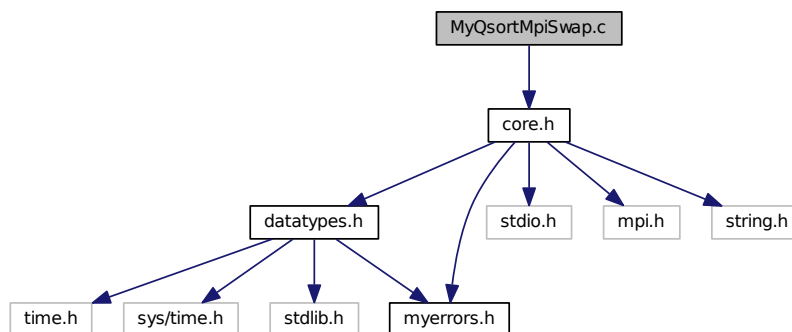
Граф вызова функции:



## 2.19 Файл MyQsortMpiSwap.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для MyQsortMpiSwap.c:



### Функции

- void [MyQsortMpiSwap](#) ([array](#) \*myArray, int ninjaIdx, int inlen, int outlen, int rank, int ProcNum, MPI\_Comm currentComm)

Обмен элементами массивов между процессами при "быстрой" сортировке

#### 2.19.1 Функции

2.19.1.1 void [MyQsortMpiSwap](#) ( array \* myArray, int ninjaIdx, int inlen, int outlen, int rank, int ProcNum, MPI\_Comm currentComm )

Обмен элементами массивов между процессами при "быстрой" сортировке

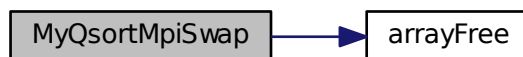
Аргументы

myArray	указатель на массив
ninjaIdx	индекс ведущего элемента
inlen	количество принимаемых элементов
outlen	количество отправляемых элементов
rank	номер процесса
ProcNum	общее число процессов
currentComm	коммуникатор

Автор

Arthur Asylgareev (Virid Raven)

Граф вызовов:



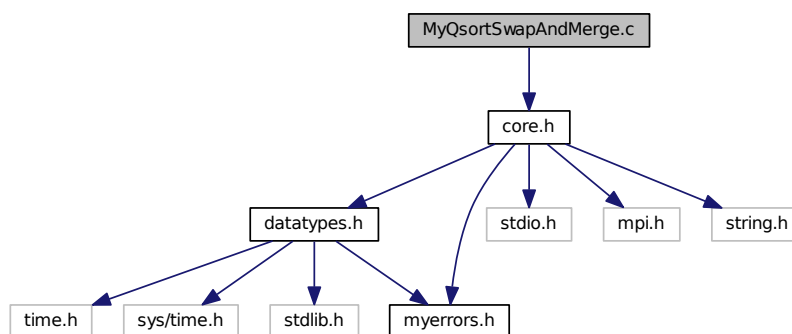
Граф вызова функции:



## 2.20 Файл MyQsortSwapAndMerge.c

```
#include "core.h"
```

Граф включаемых заголовочных файлов для MyQsortSwapAndMerge.c:



Функции

- void `MyQsortSwapAndMerge` (`array` \*myArray, `ninja` \*myNinja, int rank, int ProcNum, MPI\_Comm currentComm)

Обмен и слияние массивов между процессами при "быстрой" сортировке

### 2.20.1 Функции



```
2.20.1.1 void MyQsortSwapAndMerge ( array * myArray, ninja * myNinja, int rank, int ProcNum,  
MPI_Comm currentComm )
```

Обмен и слияние массивов между процессами при "быстрой" сортировке

## Аргументы

myArray	указатель на массив
myNinja	указатель на ниндзю
rank	номер процесса
ProcNum	общее число процессов
currentComm	коммуникатор

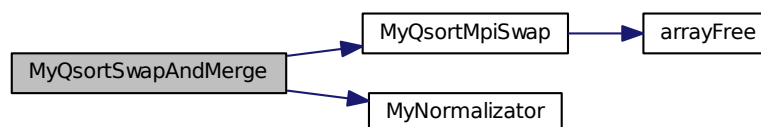
## Автор

Arthur Asylgareev (Virid Raven)

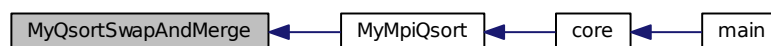
См. также

[MyMpiQsort\(\)](#)

## Граф вызовов:



## Граф вызова функции:



# Предметный указатель

- ARRAY\_INIT
  - datatypes.h, 19
- Array, 17
- array
  - datatypes.h, 20
- arrayFree
  - arrayFree.c, 3
  - datatypes.h, 20
- arrayFree.c, 3
  - arrayFree, 3
- arrayInit
  - arrayInit.c, 4
  - datatypes.h, 20
- arrayInit.c, 4
  - arrayInit, 4
- Border, 17
- border
  - datatypes.h, 20
- core
  - core.c, 6
  - core.h, 8
- core.c, 6
  - core, 6
- core.h, 7
  - core, 8
  - getAvg, 9
  - globalIsSorted, 10
  - isPowerOfTwo, 11
  - MY\_MPI\_QSORT, 8
  - MyEveSwapAndMerge, 11
  - MyMpiEVESort, 12
  - MyMpiQsort, 12
  - MyQsortMpiSwap, 13
  - MyQsortSwapAndMerge, 14
- DEFINED\_RANDOM\_MODE
  - datatypes.h, 19
- datatypes.h, 15
  - ARRAY\_INIT, 19
  - array, 20
  - arrayFree, 20
  - arrayInit, 20
  - border, 20
  - DEFINED\_RANDOM\_MODE, 19
  - fMax, 19
  - fMin, 19
  - fRand, 21
  - getData, 22
- getNinja, 22
- getNinjaIdx, 23
- getNum, 23
- getSum, 24
- isSorted, 24
- MyBubbleSort, 25
- MyNormalizator, 25
- NINJA\_INIT, 19
- ninja, 20
- RANDOM\_MODE, 19
- REPORT\_INIT, 20
- report, 20
- errorString
  - myerrors.c, 40
  - myerrors.h, 41
- fMax
  - datatypes.h, 19
- fMin
  - datatypes.h, 19
- fRand
  - datatypes.h, 21
- getAvg
  - core.h, 9
- getData
  - datatypes.h, 22
  - getData.c, 27
- getData.c, 26
  - getData, 27
- getNinja
  - datatypes.h, 22
  - getNinja.c, 28
- getNinja.c, 28
  - getNinja, 28
- getNinjaIdx
  - datatypes.h, 23
  - getNinjaIdx.c, 30
- getNinjaIdx.c, 29
  - getNinjaIdx, 30
- getNum
  - datatypes.h, 23
  - getNum.c, 30
- getNum.c, 30
  - getNum, 30
- getSum
  - datatypes.h, 24
- globalIsSorted
  - core.h, 10

globalIsSorted.c, 33  
 globalIsSorted.c, 32  
 globalIsSorted, 33  
 isPowerOfTwo  
     core.h, 11  
 isSorted  
     datatypes.h, 24  
 MY\_MPI\_EVEN\_N\_EVEN  
     core.h, 8  
 MY\_MPI\_QSORT  
     core.h, 8  
 main  
     main.c, 34  
 main.c, 34  
     main, 34  
     myerror, 37  
     printGlobalReport, 35  
     printReport, 35  
 MyBubbleSort  
     datatypes.h, 25  
     MyBubbleSort.c, 38  
 MyBubbleSort.c, 38  
     MyBubbleSort, 38  
 MyEveMpiSwapAndMerge.c, 42  
     MyEveSwapAndMerge, 42  
 MyEveSwapAndMerge  
     core.h, 11  
     MyEveMpiSwapAndMerge.c, 42  
 MyMpiEVESort  
     core.h, 12  
     MyMpiEVESort.c, 45  
 MyMpiEVESort.c, 44  
     MyMpiEVESort, 45  
 MyMpiQsort  
     core.h, 12  
     MyMpiQsort.c, 46  
 MyMpiQsort.c, 46  
     MyMpiQsort, 46  
 MyNormalizator  
     datatypes.h, 25  
     MyNormalizator.c, 48  
 MyNormalizator.c, 48  
     MyNormalizator, 48  
 MyQsortMpiSwap  
     core.h, 13  
     MyQsortMpiSwap.c, 49  
 MyQsortMpiSwap.c, 49  
     MyQsortMpiSwap, 49  
 MyQsortSwapAndMerge  
     core.h, 14  
     MyQsortSwapAndMerge.c, 50  
 MyQsortSwapAndMerge.c, 50  
     MyQsortSwapAndMerge, 50  
 myerror  
     main.c, 37  
     myerrors.h, 42  
 myerrors.c, 39

errorString, 40  
 SORT\_SUC\_STR, 40  
 SORT\_UNSUC\_STR, 40  
 UNKNOWN\_ERROR, 40  
 UNKNOWN\_MODE\_STR, 40  
 WRONG\_PROC\_NUM\_STR, 40  
 myerrors.h, 40  
     errorString, 41  
     myerror, 42  
     SORT\_SUCCEEDED, 41  
     SORT\_UNSUCCESSED, 41  
     UNKNOWN\_MODE, 41  
     WRONG\_PROC\_NUMBER, 41  
 NINJA\_INIT  
     datatypes.h, 19  
 Ninja, 18  
 ninja  
     datatypes.h, 20  
 printGlobalReport  
     main.c, 35  
 printReport  
     main.c, 35  
 RANDOM\_MODE  
     datatypes.h, 19  
 REPORT\_INIT  
     datatypes.h, 20  
 Report, 19  
 report  
     datatypes.h, 20  
 SORT\_SUC\_STR  
     myerrors.c, 40  
 SORT\_SUCCEEDED  
     myerrors.h, 41  
 SORT\_UNSUC\_STR  
     myerrors.c, 40  
 SORT\_UNSUCCESSED  
     myerrors.h, 41  
 UNKNOWN\_ERROR  
     myerrors.c, 40  
 UNKNOWN\_MODE  
     myerrors.h, 41  
 UNKNOWN\_MODE\_STR  
     myerrors.c, 40  
 WRONG\_PROC\_NUM\_STR  
     myerrors.c, 40  
 WRONG\_PROC\_NUMBER  
     myerrors.h, 41