

OP3

Generated by Doxygen 1.14.0

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Studentas Class Reference	7
4.1.1 Constructor & Destructor Documentation	8
4.1.1.1 Studentas() [1/4]	8
4.1.1.2 Studentas() [2/4]	8
4.1.1.3 ~Studentas()	8
4.1.1.4 Studentas() [3/4]	8
4.1.1.5 Studentas() [4/4]	8
4.1.2 Member Function Documentation	8
4.1.2.1 egzaminas()	8
4.1.2.2 galutinisMediana()	9
4.1.2.3 galutinisVidurkis()	9
4.1.2.4 info()	9
4.1.2.5 namuDarbai()	9
4.1.2.6 operator=() [1/2]	9
4.1.2.7 operator=() [2/2]	9
4.1.2.8 pavarde()	9
4.1.2.9 setEgzaminas()	9
4.1.2.10 skaiciuotiCache()	9
4.1.2.11 vardas()	10
4.1.3 Friends And Related Symbol Documentation	10
4.1.3.1 operator<<	10
4.1.3.2 operator>>	10
4.2 Vector< T > Class Template Reference	10
4.2.1 Member Typedef Documentation	12
4.2.1.1 const_iterator	12
4.2.1.2 const_pointer	12
4.2.1.3 const_reference	12
4.2.1.4 const_reverse_iterator	12
4.2.1.5 iterator	12
4.2.1.6 pointer	12
4.2.1.7 reference	12
4.2.1.8 reverse_iterator	13
4.2.1.9 size_type	13

4.2.1.10 value_type	13
4.2.2 Constructor & Destructor Documentation	13
4.2.2.1 Vector() [1/5]	13
4.2.2.2 Vector() [2/5]	13
4.2.2.3 Vector() [3/5]	13
4.2.2.4 Vector() [4/5]	13
4.2.2.5 Vector() [5/5]	13
4.2.2.6 ~Vector()	14
4.2.3 Member Function Documentation	14
4.2.3.1 assign() [1/3]	14
4.2.3.2 assign() [2/3]	14
4.2.3.3 assign() [3/3]	14
4.2.3.4 at() [1/2]	14
4.2.3.5 at() [2/2]	14
4.2.3.6 back() [1/2]	14
4.2.3.7 back() [2/2]	15
4.2.3.8 begin() [1/2]	15
4.2.3.9 begin() [2/2]	15
4.2.3.10 capacity()	15
4.2.3.11 clear()	15
4.2.3.12 data() [1/2]	15
4.2.3.13 data() [2/2]	15
4.2.3.14 emplace()	15
4.2.3.15 emplace_back()	16
4.2.3.16 empty()	16
4.2.3.17 end() [1/2]	16
4.2.3.18 end() [2/2]	16
4.2.3.19 erase() [1/2]	16
4.2.3.20 erase() [2/2]	16
4.2.3.21 front() [1/2]	16
4.2.3.22 front() [2/2]	16
4.2.3.23 get_allocator()	17
4.2.3.24 insert() [1/3]	17
4.2.3.25 insert() [2/3]	17
4.2.3.26 insert() [3/3]	17
4.2.3.27 operator!=(())	17
4.2.3.28 operator<()	17
4.2.3.29 operator<=()	17
4.2.3.30 operator=() [1/2]	18
4.2.3.31 operator=() [2/2]	18
4.2.3.32 operator==(())	18
4.2.3.33 operator>()	18

4.2.3.34 operator>=()	18
4.2.3.35 operator[]() [1/2]	18
4.2.3.36 operator[]() [2/2]	18
4.2.3.37 pop_back()	18
4.2.3.38 push_back() [1/2]	19
4.2.3.39 push_back() [2/2]	19
4.2.3.40 rbegin() [1/2]	19
4.2.3.41 rbegin() [2/2]	19
4.2.3.42 rend() [1/2]	19
4.2.3.43 rend() [2/2]	19
4.2.3.44 reserve()	19
4.2.3.45 resize()	19
4.2.3.46 shrink_to_fit()	20
4.2.3.47 size()	20
4.2.3.48 swap()	20
4.3 Zmogus Class Reference	20
4.3.1 Constructor & Destructor Documentation	21
4.3.1.1 Zmogus() [1/2]	21
4.3.1.2 Zmogus() [2/2]	21
4.3.1.3 ~Zmogus()	21
4.3.2 Member Function Documentation	21
4.3.2.1 info()	21
4.3.2.2 pavarde()	21
4.3.2.3 vardas()	21
4.3.3 Member Data Documentation	21
4.3.3.1 pav_	21
4.3.3.2 var_	21
5 File Documentation	23
5.1 bench.cpp File Reference	23
5.1.1 Function Documentation	23
5.1.1.1 main()	23
5.2 main.cpp File Reference	23
5.2.1 Function Documentation	24
5.2.1.1 main()	24
5.2.1.2 promptForSortingMethod()	24
5.2.2 Variable Documentation	24
5.2.2.1 MAX_STUDENTU_SKAICIUS	24
5.2.2.2 rikiavimas	24
5.3 Mylib.h File Reference	24
5.4 Mylib.h	25
5.5 Studentas.cpp File Reference	25

5.5.1 Function Documentation	26
5.5.1.1 clearInput()	26
5.5.1.2 generateStudentFiles()	26
5.5.1.3 generuotiPavarde()	26
5.5.1.4 generuotiVarda()	26
5.5.1.5 handleFileInput()	26
5.5.1.6 handleOutput()	26
5.5.1.7 iverstiStudentus()	27
5.5.1.8 Mediana()	27
5.5.1.9 operator<<()	27
5.5.1.10 operator>>()	27
5.5.1.11 rikiuotiStudentus()	27
5.5.1.12 skaitymas()	27
5.5.1.13 sortAndOutputStudents()	27
5.5.1.14 spausdinti()	27
5.5.2 Variable Documentation	28
5.5.2.1 rikiavimas	28
5.6 Studentas.h File Reference	28
5.6.1 Function Documentation	28
5.6.1.1 clearInput()	28
5.6.1.2 generateStudentFiles()	29
5.6.1.3 generuotiPavarde()	29
5.6.1.4 generuotiVarda()	29
5.6.1.5 handleFileInput()	29
5.6.1.6 handleOutput()	29
5.6.1.7 iverstiStudentus()	29
5.6.1.8 Mediana()	29
5.6.1.9 rikiuotiStudentus()	29
5.6.1.10 skaitymas()	29
5.6.1.11 sortAndOutputStudents()	30
5.6.1.12 spausdinti()	30
5.7 Studentas.h	30
5.8 testStudentas.cpp File Reference	31
5.8.1 Function Documentation	31
5.8.1.1 main()	31
5.8.1.2 spausdintiStudenta()	31
5.9 Vector.h File Reference	31
5.9.1 Function Documentation	32
5.9.1.1 swap()	32
5.10 Vector.h	32
5.11 Zmogus.h File Reference	35
5.12 Zmogus.h	35

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Vector< T >	10
Zmogus	20
Studentas	7

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Studentas	7
Vector< T >	10
Zmogus	20

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

bench.cpp	23
main.cpp	23
Mylib.h	24
Studentas.cpp	25
Studentas.h	28
testStudentas.cpp	31
Vector.h	31
Zmogus.h	35

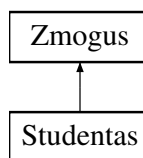
Chapter 4

Class Documentation

4.1 Studentas Class Reference

```
#include <Studentas.h>
```

Inheritance diagram for Studentas:



Public Member Functions

- [Studentas](#) ()
- [Studentas](#) (const std::string &var, const std::string &pav, const [Vector](#)< int > &nd, int egz)
- [~Studentas](#) ()
- [Studentas](#) (const [Studentas](#) &other)
- [Studentas](#) ([Studentas](#) &&other) noexcept
- [Studentas](#) & [operator=](#) (const [Studentas](#) &other)
- [Studentas](#) & [operator=](#) ([Studentas](#) &&other) noexcept
- std::string [vardas](#) () const override
- std::string [pavarde](#) () const override
- [Vector](#)< int > [namuDarbai](#) () const
- int [egzaminas](#) () const
- void [setEgzaminas](#) (int egz)
- double [galutinisVidurkis](#) () const
- double [galutinisMediana](#) () const
- void [skaiciuotiCache](#) () const
- void [info](#) () const override

Public Member Functions inherited from [Zmogus](#)

- [Zmogus](#) ()
- [Zmogus](#) (const std::string &var, const std::string &pav)
- virtual [~Zmogus](#) ()=default

Friends

- `std::istream & operator>>` (`std::istream &in`, `Studentas &s`)
- `std::ostream & operator<<` (`std::ostream &out`, `const Studentas &s`)

Additional Inherited Members

Protected Attributes inherited from `Zmogus`

- `std::string var_`
- `std::string pav_`

4.1.1 Constructor & Destructor Documentation

4.1.1.1 `Studentas()` [1/4]

```
Studentas::Studentas ()
```

4.1.1.2 `Studentas()` [2/4]

```
Studentas::Studentas (  
    const std::string & var,  
    const std::string & pav,  
    const Vector< int > & nd,  
    int egz)
```

4.1.1.3 `~Studentas()`

```
Studentas::~~Studentas ()
```

4.1.1.4 `Studentas()` [3/4]

```
Studentas::Studentas (  
    const Studentas & other)
```

4.1.1.5 `Studentas()` [4/4]

```
Studentas::Studentas (  
    Studentas && other) [noexcept]
```

4.1.2 Member Function Documentation

4.1.2.1 `egzaminas()`

```
int Studentas::egzaminas () const [inline]
```


4.1.2.2 galutinisMediana()

```
double Studentas::galutinisMediana () const
```

4.1.2.3 galutinisVidurkis()

```
double Studentas::galutinisVidurkis () const
```

4.1.2.4 info()

```
void Studentas::info () const [inline], [override], [virtual]
```

Implements [Zmogus](#).

4.1.2.5 namuDarbai()

```
Vector< int > Studentas::namuDarbai () const [inline]
```

4.1.2.6 operator=() [1/2]

```
Studentas & Studentas::operator= (
    const Studentas & other)
```

4.1.2.7 operator=() [2/2]

```
Studentas & Studentas::operator= (
    Studentas && other) [noexcept]
```

4.1.2.8 pavarde()

```
std::string Studentas::pavarde () const [inline], [override], [virtual]
```

Reimplemented from [Zmogus](#).

4.1.2.9 setEgzaminas()

```
void Studentas::setEgzaminas (
    int egz) [inline]
```

4.1.2.10 skaiciuotiCache()

```
void Studentas::skaiciuotiCache () const
```

4.1.2.11 vardas()

```
std::string Studentas::vardas () const [inline], [override], [virtual]
```

Reimplemented from [Zmogus](#).

4.1.3 Friends And Related Symbol Documentation

4.1.3.1 operator<<

```
std::ostream & operator<< (  
    std::ostream & out,  
    const Studentas & s) [friend]
```

4.1.3.2 operator>>

```
std::istream & operator>> (  
    std::istream & in,  
    Studentas & s) [friend]
```

The documentation for this class was generated from the following files:

- [Studentas.h](#)
- [Studentas.cpp](#)

4.2 Vector< T > Class Template Reference

```
#include <Vector.h>
```

Public Types

- using [value_type](#) = T
- using [size_type](#) = std::size_t
- using [reference](#) = T&
- using [const_reference](#) = const T&
- using [pointer](#) = T*
- using [const_pointer](#) = const T*
- using [iterator](#) = T*
- using [const_iterator](#) = const T*
- using [reverse_iterator](#) = std::reverse_iterator<[iterator](#)>
- using [const_reverse_iterator](#) = std::reverse_iterator<[const_iterator](#)>

Public Member Functions

- [Vector](#) ()
- [Vector](#) ([size_type](#) n, const T &val=T())
- [Vector](#) (std::initializer_list< T > il)
- [Vector](#) (const [Vector](#) &other)
- [Vector](#) ([Vector](#) &&other) noexcept
- [Vector](#) & [operator=](#) (const [Vector](#) &other)
- [Vector](#) & [operator=](#) ([Vector](#) &&other) noexcept
- [~Vector](#) ()
- [reference operator\[\]](#) ([size_type](#) i)
- [const_reference operator\[\]](#) ([size_type](#) i) const
- [reference at](#) ([size_type](#) i)
- [const_reference at](#) ([size_type](#) i) const
- [reference front](#) ()
- [const_reference front](#) () const
- [reference back](#) ()
- [const_reference back](#) () const
- [pointer data](#) () noexcept
- [const_pointer data](#) () const noexcept
- [iterator begin](#) () noexcept
- [const_iterator begin](#) () const noexcept
- [iterator end](#) () noexcept
- [const_iterator end](#) () const noexcept
- [reverse_iterator rbegin](#) () noexcept
- [const_reverse_iterator rbegin](#) () const noexcept
- [reverse_iterator rend](#) () noexcept
- [const_reverse_iterator rend](#) () const noexcept
- bool [empty](#) () const noexcept
- [size_type size](#) () const noexcept
- [size_type capacity](#) () const noexcept
- void [reserve](#) ([size_type](#) n)
- void [shrink_to_fit](#) ()
- void [clear](#) () noexcept
- void [push_back](#) (const T &val)
- void [push_back](#) (T &&val)
- void [pop_back](#) ()
- void [resize](#) ([size_type](#) n, const T &val=T())
- void [swap](#) ([Vector](#) &other) noexcept
- [iterator insert](#) ([const_iterator](#) pos, const T &value)
- [iterator erase](#) ([const_iterator](#) pos)
- void [assign](#) ([size_type](#) n, const T &value)
- void [assign](#) (std::initializer_list< T > il)
- template<typename InputIt>
std::enable_if<!std::is_integral< InputIt >::value, void >::type [assign](#) (InputIt first, InputIt last)
- template<typename... Args>
void [emplace_back](#) (Args &&... args)
- template<typename InputIt>
[iterator insert](#) ([const_iterator](#) pos, InputIt first, InputIt last)
- [iterator insert](#) ([const_iterator](#) pos, std::initializer_list< T > ilist)
- [iterator erase](#) ([const_iterator](#) first, [const_iterator](#) last)
- template<typename... Args>
[iterator emplace](#) ([const_iterator](#) pos, Args &&... args)
- std::allocator< T > [get_allocator](#) () const
- bool [operator==](#) (const [Vector](#) &other) const

- bool `operator!=` (const `Vector` &other) const
- bool `operator<` (const `Vector` &other) const
- bool `operator>` (const `Vector` &other) const
- bool `operator<=` (const `Vector` &other) const
- bool `operator>=` (const `Vector` &other) const

4.2.1 Member Typedef Documentation

4.2.1.1 `const_iterator`

```
template<typename T>  
using Vector< T >::const_iterator = const T*
```

4.2.1.2 `const_pointer`

```
template<typename T>  
using Vector< T >::const_pointer = const T*
```

4.2.1.3 `const_reference`

```
template<typename T>  
using Vector< T >::const_reference = const T&
```

4.2.1.4 `const_reverse_iterator`

```
template<typename T>  
using Vector< T >::const_reverse_iterator = std::reverse_iterator<const_iterator>
```

4.2.1.5 `iterator`

```
template<typename T>  
using Vector< T >::iterator = T*
```

4.2.1.6 `pointer`

```
template<typename T>  
using Vector< T >::pointer = T*
```

4.2.1.7 `reference`

```
template<typename T>  
using Vector< T >::reference = T&
```

4.2.1.8 reverse_iterator

```
template<typename T>
using Vector< T >::reverse_iterator = std::reverse_iterator<iterator>
```

4.2.1.9 size_type

```
template<typename T>
using Vector< T >::size_type = std::size_t
```

4.2.1.10 value_type

```
template<typename T>
using Vector< T >::value_type = T
```

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Vector() [1/5]

```
template<typename T>
Vector< T >::Vector () [inline]
```

4.2.2.2 Vector() [2/5]

```
template<typename T>
Vector< T >::Vector (
    size_type n,
    const T & val = T()) [inline]
```

4.2.2.3 Vector() [3/5]

```
template<typename T>
Vector< T >::Vector (
    std::initializer_list< T > il) [inline]
```

4.2.2.4 Vector() [4/5]

```
template<typename T>
Vector< T >::Vector (
    const Vector< T > & other) [inline]
```

4.2.2.5 Vector() [5/5]

```
template<typename T>
Vector< T >::Vector (
    Vector< T > && other) [inline], [noexcept]
```

4.2.2.6 ~Vector()

```
template<typename T>
Vector< T >::~~Vector () [inline]
```

4.2.3 Member Function Documentation

4.2.3.1 assign() [1/3]

```
template<typename T>
template<typename InputIt>
std::enable_if<!std::is_integral< InputIt >::value, void >::type Vector< T >::assign (
    InputIt first,
    InputIt last) [inline]
```

4.2.3.2 assign() [2/3]

```
template<typename T>
void Vector< T >::assign (
    size_type n,
    const T & value) [inline]
```

4.2.3.3 assign() [3/3]

```
template<typename T>
void Vector< T >::assign (
    std::initializer_list< T > il) [inline]
```

4.2.3.4 at() [1/2]

```
template<typename T>
reference Vector< T >::at (
    size_type i) [inline]
```

4.2.3.5 at() [2/2]

```
template<typename T>
const_reference Vector< T >::at (
    size_type i) const [inline]
```

4.2.3.6 back() [1/2]

```
template<typename T>
reference Vector< T >::back () [inline]
```

4.2.3.7 back() [2/2]

```
template<typename T>
const_reference Vector< T >::back () const [inline]
```

4.2.3.8 begin() [1/2]

```
template<typename T>
const_iterator Vector< T >::begin () const [inline], [noexcept]
```

4.2.3.9 begin() [2/2]

```
template<typename T>
iterator Vector< T >::begin () [inline], [noexcept]
```

4.2.3.10 capacity()

```
template<typename T>
size_type Vector< T >::capacity () const [inline], [noexcept]
```

4.2.3.11 clear()

```
template<typename T>
void Vector< T >::clear () [inline], [noexcept]
```

4.2.3.12 data() [1/2]

```
template<typename T>
const_pointer Vector< T >::data () const [inline], [noexcept]
```

4.2.3.13 data() [2/2]

```
template<typename T>
pointer Vector< T >::data () [inline], [noexcept]
```

4.2.3.14 emplace()

```
template<typename T>
template<typename... Args>
iterator Vector< T >::emplace (
    const_iterator pos,
    Args &&... args) [inline]
```

4.2.3.15 `emplace_back()`

```
template<typename T>
template<typename... Args>
void Vector< T >::emplace_back (
    Args &&... args) [inline]
```

4.2.3.16 `empty()`

```
template<typename T>
bool Vector< T >::empty () const [inline], [noexcept]
```

4.2.3.17 `end()` [1/2]

```
template<typename T>
const_iterator Vector< T >::end () const [inline], [noexcept]
```

4.2.3.18 `end()` [2/2]

```
template<typename T>
iterator Vector< T >::end () [inline], [noexcept]
```

4.2.3.19 `erase()` [1/2]

```
template<typename T>
iterator Vector< T >::erase (
    const_iterator first,
    const_iterator last) [inline]
```

4.2.3.20 `erase()` [2/2]

```
template<typename T>
iterator Vector< T >::erase (
    const_iterator pos) [inline]
```

4.2.3.21 `front()` [1/2]

```
template<typename T>
reference Vector< T >::front () [inline]
```

4.2.3.22 `front()` [2/2]

```
template<typename T>
const_reference Vector< T >::front () const [inline]
```


4.2.3.23 get_allocator()

```
template<typename T>
std::allocator< T > Vector< T >::get_allocator () const [inline]
```

4.2.3.24 insert() [1/3]

```
template<typename T>
iterator Vector< T >::insert (
    const_iterator pos,
    const T & value) [inline]
```

4.2.3.25 insert() [2/3]

```
template<typename T>
template<typename InputIt>
iterator Vector< T >::insert (
    const_iterator pos,
    InputIt first,
    InputIt last) [inline]
```

4.2.3.26 insert() [3/3]

```
template<typename T>
iterator Vector< T >::insert (
    const_iterator pos,
    std::initializer_list< T > ilist) [inline]
```

4.2.3.27 operator"!="()

```
template<typename T>
bool Vector< T >::operator!= (
    const Vector< T > & other) const [inline]
```

4.2.3.28 operator<()

```
template<typename T>
bool Vector< T >::operator< (
    const Vector< T > & other) const [inline]
```

4.2.3.29 operator<=()

```
template<typename T>
bool Vector< T >::operator<= (
    const Vector< T > & other) const [inline]
```

4.2.3.30 operator=() [1/2]

```
template<typename T>
Vector & Vector< T >::operator= (
    const Vector< T > & other) [inline]
```

4.2.3.31 operator=() [2/2]

```
template<typename T>
Vector & Vector< T >::operator= (
    Vector< T > && other) [inline], [noexcept]
```

4.2.3.32 operator==()

```
template<typename T>
bool Vector< T >::operator== (
    const Vector< T > & other) const [inline]
```

4.2.3.33 operator>()

```
template<typename T>
bool Vector< T >::operator> (
    const Vector< T > & other) const [inline]
```

4.2.3.34 operator>=()

```
template<typename T>
bool Vector< T >::operator>= (
    const Vector< T > & other) const [inline]
```

4.2.3.35 operator[]() [1/2]

```
template<typename T>
reference Vector< T >::operator[] (
    size_type i) [inline]
```

4.2.3.36 operator[]() [2/2]

```
template<typename T>
const_reference Vector< T >::operator[] (
    size_type i) const [inline]
```

4.2.3.37 pop_back()

```
template<typename T>
void Vector< T >::pop_back () [inline]
```

4.2.3.38 push_back() [1/2]

```
template<typename T>
void Vector< T >::push_back (
    const T & val) [inline]
```

4.2.3.39 push_back() [2/2]

```
template<typename T>
void Vector< T >::push_back (
    T && val) [inline]
```

4.2.3.40 rbegin() [1/2]

```
template<typename T>
const_reverse_iterator Vector< T >::rbegin () const [inline], [noexcept]
```

4.2.3.41 rbegin() [2/2]

```
template<typename T>
reverse_iterator Vector< T >::rbegin () [inline], [noexcept]
```

4.2.3.42 rend() [1/2]

```
template<typename T>
const_reverse_iterator Vector< T >::rend () const [inline], [noexcept]
```

4.2.3.43 rend() [2/2]

```
template<typename T>
reverse_iterator Vector< T >::rend () [inline], [noexcept]
```

4.2.3.44 reserve()

```
template<typename T>
void Vector< T >::reserve (
    size_type n) [inline]
```

4.2.3.45 resize()

```
template<typename T>
void Vector< T >::resize (
    size_type n,
    const T & val = T()) [inline]
```

4.2.3.46 shrink_to_fit()

```
template<typename T>
void Vector< T >::shrink_to_fit () [inline]
```

4.2.3.47 size()

```
template<typename T>
size_type Vector< T >::size () const [inline], [noexcept]
```

4.2.3.48 swap()

```
template<typename T>
void Vector< T >::swap (
    Vector< T > & other) [inline], [noexcept]
```

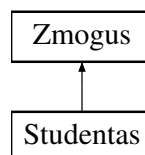
The documentation for this class was generated from the following file:

- [Vector.h](#)

4.3 Zmogus Class Reference

```
#include <Zmogus.h>
```

Inheritance diagram for Zmogus:



Public Member Functions

- [Zmogus](#) ()
- [Zmogus](#) (const std::string &var, const std::string &pav)
- virtual [~Zmogus](#) ()=default
- virtual std::string [vardas](#) () const
- virtual std::string [pavarde](#) () const
- virtual void [info](#) () const =0

Protected Attributes

- std::string [var_](#)
- std::string [pav_](#)

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Zmogus() [1/2]

```
Zmogus::Zmogus () [inline]
```

4.3.1.2 Zmogus() [2/2]

```
Zmogus::Zmogus (  
    const std::string & var,  
    const std::string & pav) [inline]
```

4.3.1.3 ~Zmogus()

```
virtual Zmogus::~Zmogus () [virtual], [default]
```

4.3.2 Member Function Documentation

4.3.2.1 info()

```
virtual void Zmogus::info () const [pure virtual]
```

Implemented in [Studentas](#).

4.3.2.2 pavarde()

```
virtual std::string Zmogus::pavarde () const [inline], [virtual]
```

Reimplemented in [Studentas](#).

4.3.2.3 vardas()

```
virtual std::string Zmogus::vardas () const [inline], [virtual]
```

Reimplemented in [Studentas](#).

4.3.3 Member Data Documentation

4.3.3.1 pav_

```
std::string Zmogus::pav_ [protected]
```

4.3.3.2 var_

```
std::string Zmogus::var_ [protected]
```

The documentation for this class was generated from the following file:

- [Zmogus.h](#)

Chapter 5

File Documentation

5.1 bench.cpp File Reference

```
#include <iostream>
#include <vector>
#include <chrono>
#include "Vector.h"
#include <iomanip>
```

Functions

- int `main` ()

5.1.1 Function Documentation

5.1.1.1 `main()`

```
int main ()
```

5.2 main.cpp File Reference

```
#include "Studentas.h"
#include "Mylib.h"
#include <iostream>
#include "Vector.h"
#include <chrono>
#include <ctime>
```

Functions

- void `promptForSortingMethod` ()
- int `main` ()

Variables

- char [rikiavimas](#)
- const int [MAX_STUDENTU_SKAICIUS](#) = 10000000

5.2.1 Function Documentation

5.2.1.1 main()

```
int main ()
```

5.2.1.2 promptForSortingMethod()

```
void promptForSortingMethod ()
```

5.2.2 Variable Documentation

5.2.2.1 MAX_STUDENTU_SKAICIUS

```
const int MAX_STUDENTU_SKAICIUS = 10000000
```

5.2.2.2 rikiavimas

```
char rikiavimas [extern]
```

5.3 Mylib.h File Reference

```
#include <iostream>
#include <string>
#include <iomanip>
#include <algorithm>
#include "Vector.h"
#include <limits>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <sstream>
#include <chrono>
#include <stdexcept>
#include <list>
```


5.4 Mylib.h

[Go to the documentation of this file.](#)

```
00001 #ifndef MYLIB_H
00002 #define MYLIB_H
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <iomanip>
00007 #include <algorithm>
00008 #include "Vector.h"
00009 #include <limits>
00010 #include <cstdlib>
00011 #include <ctime>
00012 #include <fstream>
00013 #include <sstream>
00014 #include <chrono>
00015 #include <stdexcept>
00016 #include <list>
00017
00018 using std::string;
00019 using std::cout;
00020 using std::cin;
00021 using std::endl;
00022 using std::fixed;
00023 using std::setprecision;
00024 using std::setw;
00025 using std::left;
00026 using std::vector;
00027 using std::ifstream;
00028 using std::ofstream;
00029 using std::stringstream;
00030 using std::chrono::high_resolution_clock;
00031 using std::chrono::duration_cast;
00032 using std::chrono::duration;
00033 using std::chrono::seconds;
00034 using std::to_string;
00035 using std::exception;
00036
00037 #endif // MYLIB_H
```

5.5 Studentas.cpp File Reference

```
#include "Studentas.h"
#include "Mylib.h"
#include <fstream>
#include <iomanip>
#include <iostream>
#include "Vector.h"
#include <chrono>
#include <ctime>
#include <sstream>
#include <stdexcept>
#include <limits>
#include <algorithm>
#include <direct.h>
```

Functions

- `std::istream & operator>>` (`std::istream &in`, `Studentas &s`)
- `std::ostream & operator<<` (`std::ostream &out`, `const Studentas &s`)
- `double Mediana` (`const Vector< int > &vec`)
- `void clearInput` ()
- `string generuotiVarda` ()

- string `generuotiPavarde` ()
- bool `skaitymas` (`Vector< Studentas >` &studentai, const string &failoPav)
- void `spausdinti` (const `Vector< Studentas >` &studentai, ostream &out)
- void `rikiuotiStudentus` (`Vector< Studentas >` &studentai, char `rikiavimas`)
- void `ivestiStudentus` (`Vector< Studentas >` &studentai)
- void `sortAndOutputStudents` (`Vector< Studentas >` &studentai)
- void `handleFileInput` (`Vector< Studentas >` &studentai)
- void `handleOutput` (const `Vector< Studentas >` &studentai)
- void `generateStudentFiles` ()

Variables

- char `rikiavimas`

5.5.1 Function Documentation

5.5.1.1 clearInput()

```
void clearInput ()
```

5.5.1.2 generateStudentFiles()

```
void generateStudentFiles ()
```

5.5.1.3 generuotiPavarde()

```
string generuotiPavarde ()
```

5.5.1.4 generuotiVarda()

```
string generuotiVarda ()
```

5.5.1.5 handleFileInput()

```
void handleFileInput (  
    Vector< Studentas > & studentai)
```

5.5.1.6 handleOutput()

```
void handleOutput (  
    const Vector< Studentas > & studentai)
```

5.5.1.7 iverstiStudentus()

```
void iverstiStudentus (
    Vector< Studentas > & studentai)
```

5.5.1.8 Mediana()

```
double Mediana (
    const Vector< int > & vec)
```

5.5.1.9 operator<<()

```
std::ostream & operator<< (
    std::ostream & out,
    const Studentas & s)
```

5.5.1.10 operator>>()

```
std::istream & operator>> (
    std::istream & in,
    Studentas & s)
```

5.5.1.11 rikiuotiStudentus()

```
void rikiuotiStudentus (
    Vector< Studentas > & studentai,
    char rikiavimas)
```

5.5.1.12 skaitymas()

```
bool skaitymas (
    Vector< Studentas > & studentai,
    const string & failoPav)
```

5.5.1.13 sortAndOutputStudents()

```
void sortAndOutputStudents (
    Vector< Studentas > & studentai)
```

5.5.1.14 spausdinti()

```
void spausdinti (
    const Vector< Studentas > & studentai,
    ostream & out)
```

5.5.2 Variable Documentation

5.5.2.1 rikiavimas

```
char rikiavimas
```

5.6 Studentas.h File Reference

```
#include <iostream>
#include "Vector.h"
#include <string>
#include <algorithm>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <chrono>
#include <limits>
#include "Zmogus.h"
```

Classes

- class [Studentas](#)

Functions

- double [Mediana](#) (const [Vector](#)< int > &vec)
- void [clearInput](#) ()
- std::string [generuotiVarda](#) ()
- std::string [generuotiPavarde](#) ()
- bool [skaitymas](#) ([Vector](#)< [Studentas](#) > &studentai, const std::string &failoPav)
- void [spausdinti](#) (const [Vector](#)< [Studentas](#) > &studentai, std::ostream &out)
- void [rikiuotiStudentus](#) ([Vector](#)< [Studentas](#) > &studentai, char [rikiavimas](#))
- void [ivestiStudentus](#) ([Vector](#)< [Studentas](#) > &studentai)
- void [sortAndOutputStudents](#) ([Vector](#)< [Studentas](#) > &studentai)
- void [handleFileInput](#) ([Vector](#)< [Studentas](#) > &studentai)
- void [handleOutput](#) (const [Vector](#)< [Studentas](#) > &studentai)
- void [generateStudentFiles](#) ()

5.6.1 Function Documentation

5.6.1.1 clearInput()

```
void clearInput ()
```

5.6.1.2 generateStudentFiles()

```
void generateStudentFiles ()
```

5.6.1.3 generuotiPavarde()

```
std::string generuotiPavarde ()
```

5.6.1.4 generuotiVarda()

```
std::string generuotiVarda ()
```

5.6.1.5 handleFileInput()

```
void handleFileInput (  
    Vector< Studentas > & studentai)
```

5.6.1.6 handleOutput()

```
void handleOutput (  
    const Vector< Studentas > & studentai)
```

5.6.1.7 iverstiStudentus()

```
void iverstiStudentus (  
    Vector< Studentas > & studentai)
```

5.6.1.8 Mediana()

```
double Mediana (  
    const Vector< int > & vec)
```

5.6.1.9 rikiuotiStudentus()

```
void rikiuotiStudentus (  
    Vector< Studentas > & studentai,  
    char rikiavimas)
```

5.6.1.10 skaitymas()

```
bool skaitymas (  
    Vector< Studentas > & studentai,  
    const std::string & failoPav)
```

5.6.1.11 sortAndOutputStudents()

```
void sortAndOutputStudents (
    Vector< Studentas > & studentai)
```

5.6.1.12 spausdinti()

```
void spausdinti (
    const Vector< Studentas > & studentai,
    std::ostream & out)
```

5.7 Studentas.h

[Go to the documentation of this file.](#)

```
00001 #ifndef STUDENTAS_H
00002 #define STUDENTAS_H
00003
00004 #include <iostream>
00005 #include "Vector.h"
00006 #include <string>
00007 #include <algorithm>
00008 #include <iomanip>
00009 #include <fstream>
00010 #include <sstream>
00011 #include <stdexcept>
00012 #include <chrono>
00013 #include <limits>
00014 #include "Zmogus.h"
00015
00016 class Studentas : public Zmogus {
00017 private:
00018     Vector<int> nd_;
00019     int egz_;
00020     mutable double cachedVidurkis = -1;
00021     mutable double cachedMediana = -1;
00022
00023 public:
00024     // Konstruktoriai ir destruktorius
00025     Studentas();
00026     Studentas(const std::string& var, const std::string& pav, const Vector<int>& nd, int egz);
00027     ~Studentas();
00028
00029     // Rule of Five
00030     Studentas(const Studentas& other);
00031     Studentas(Studentas&& other) noexcept;
00032     Studentas& operator=(const Studentas& other);
00033     Studentas& operator=(Studentas&& other) noexcept;
00034
00035     // Get'eriai (override)
00036     std::string vardas() const override { return var_; }
00037     std::string pavarde() const override { return pav_; }
00038     Vector<int> namuDarbai() const { return nd_; }
00039     int egzaminas() const { return egz_; }
00040
00041     // Set'eriai
00042     void setEgzaminas(int egz) { egz_ = egz; }
00043
00044     // Galutinio balo skaičiavimas
00045     double galutinisVidurkis() const;
00046     double galutinisMediana() const;
00047     void skaiciuotiCache() const;
00048
00049     // Įvesties/išvesties operatoriai
00050     friend std::istream& operator>(std::istream& in, Studentas& s);
00051     friend std::ostream& operator<(std::ostream& out, const Studentas& s);
00052
00053
00054     void info() const override {
00055         std::cout << "Studentas: " << var_ << " " << pav_ << std::endl;
00056     }
00057 };
00058
00059 // Pagalbinės funkcijos
```

```

00060 double Mediana(const Vector<int>& vec);
00061 void clearInput();
00062 std::string generuotiVarda();
00063 std::string generuotiPavarde();
00064 bool skaitymas(Vector<Studentas>& studentai, const std::string& failoPav);
00065 void spausdinti(const Vector<Studentas>& studentai, std::ostream& out);
00066 void rikiuotiStudentus(Vector<Studentas>& studentai, char rikiavimas);
00067 void ivestiStudentus(Vector<Studentas>& studentai);
00068 void sortAndOutputStudents(Vector<Studentas>& studentai);
00069 void handleFileInput(Vector<Studentas>& studentai);
00070 void handleOutput(const Vector<Studentas>& studentai);
00071 void generateStudentFiles();
00072
00073 #endif // STUDENTAS_H

```

5.8 testStudentas.cpp File Reference

```

#include "Vector.h"
#include "Studentas.h"
#include "Zmogus.h"
#include <iostream>
#include <sstream>
#include <cassert>

```

Functions

- void `spausdintiStudenta` (const `Studentas` &`s`, const string &`prefix`)
- int `main` ()

5.8.1 Function Documentation

5.8.1.1 `main()`

```
int main ()
```

5.8.1.2 `spausdintiStudenta()`

```
void spausdintiStudenta (
    const Studentas & s,
    const string & prefix)
```

5.9 Vector.h File Reference

```

#include <cstddef>
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <memory>
#include <utility>
#include <type_traits>
#include <vector>

```

Classes

- class [Vector< T >](#)

Functions

- [template<typename T>](#)
void [swap](#) ([Vector< T >](#) &a, [Vector< T >](#) &b) noexcept

5.9.1 Function Documentation

5.9.1.1 swap()

```
template<typename T>
void swap (
    Vector< T > & a,
    Vector< T > & b) [noexcept]
```

5.10 Vector.h

[Go to the documentation of this file.](#)

```
00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <stdexcept>
00007 #include <algorithm>
00008 #include <iterator>
00009 #include <memory>
00010 #include <utility>
00011 #include <type_traits>
00012 #include <vector> // tik palyginimui
00013
00014 template <typename T>
00015 class Vector {
00016 public:
00017     // Member types
00018     using value_type = T;
00019     using size_type = std::size_t;
00020     using reference = T&;
00021     using const_reference = const T&;
00022     using pointer = T*;
00023     using const_pointer = const T*;
00024     using iterator = T*;
00025     using const_iterator = const T*;
00026     using reverse_iterator = std::reverse_iterator<iterator>;
00027     using const_reverse_iterator = std::reverse_iterator<const_iterator>;
00028
00029 private:
00030     pointer data_;
00031     size_type sz_;
00032     size_type cap_;
00033
00034     void reallocate(size_type new_cap) {
00035         pointer new_data = new value_type[new_cap];
00036         for (size_type i = 0; i < sz_; ++i)
00037             new_data[i] = std::move(data_[i]);
00038         delete[] data_;
00039         data_ = new_data;
00040         cap_ = new_cap;
00041     }
00042
00043 public:
00044     // Constructors
00045     Vector() : data_(nullptr), sz_(0), cap_(0) {}
```



```

00046     Vector(size_type n, const T& val = T()) : data_(new T[n]), sz_(n), cap_(n) {
00047         std::fill(data_, data_ + n, val);
00048     }
00049     Vector(std::initializer_list<T> il) : Vector(il.size()) {
00050         std::copy(il.begin(), il.end(), data_);
00051     }
00052     Vector(const Vector& other) : data_(new T[other.cap_]), sz_(other.sz_), cap_(other.cap_) {
00053         std::copy(other.data_, other.data_ + sz_, data_);
00054     }
00055     Vector(Vector&& other) noexcept : data_(other.data_), sz_(other.sz_), cap_(other.cap_) {
00056         other.data_ = nullptr; other.sz_ = 0; other.cap_ = 0;
00057     }
00058     Vector& operator=(const Vector& other) {
00059         if (this != &other) {
00060             delete[] data_;
00061             sz_ = other.sz_;
00062             cap_ = other.cap_;
00063             data_ = new T[cap_];
00064             std::copy(other.data_, other.data_ + sz_, data_);
00065         }
00066         return *this;
00067     }
00068     Vector& operator=(Vector&& other) noexcept {
00069         if (this != &other) {
00070             delete[] data_;
00071             data_ = other.data_;
00072             sz_ = other.sz_;
00073             cap_ = other.cap_;
00074             other.data_ = nullptr; other.sz_ = 0; other.cap_ = 0;
00075         }
00076         return *this;
00077     }
00078     ~Vector() { delete[] data_; }
00079
00080     // Element access
00081     reference operator[](size_type i) { return data_[i]; }
00082     const_reference operator[](size_type i) const { return data_[i]; }
00083     reference at(size_type i) {
00084         if (i >= sz_) throw std::out_of_range("Vector::at");
00085         return data_[i];
00086     }
00087     const_reference at(size_type i) const {
00088         if (i >= sz_) throw std::out_of_range("Vector::at");
00089         return data_[i];
00090     }
00091     reference front() { return data_[0]; }
00092     const_reference front() const { return data_[0]; }
00093     reference back() { return data_[sz_ - 1]; }
00094     const_reference back() const { return data_[sz_ - 1]; }
00095     pointer data() noexcept { return data_; }
00096     const_pointer data() const noexcept { return data_; }
00097
00098     // Iterators
00099     iterator begin() noexcept { return data_; }
00100     const_iterator begin() const noexcept { return data_; }
00101     iterator end() noexcept { return data_ + sz_; }
00102     const_iterator end() const noexcept { return data_ + sz_; }
00103     reverse_iterator rbegin() noexcept { return reverse_iterator(end()); }
00104     const_reverse_iterator rbegin() const noexcept { return const_reverse_iterator(end()); }
00105     reverse_iterator rend() noexcept { return reverse_iterator(begin()); }
00106     const_reverse_iterator rend() const noexcept { return const_reverse_iterator(begin()); }
00107
00108     // Capacity
00109     bool empty() const noexcept { return sz_ == 0; }
00110     size_type size() const noexcept { return sz_; }
00111     size_type capacity() const noexcept { return cap_; }
00112     void reserve(size_type n) {
00113         if (n > cap_) reallocate(n);
00114     }
00115     void shrink_to_fit() {
00116         if (sz_ < cap_) reallocate(sz_);
00117     }
00118
00119     // Modifiers
00120     void clear() noexcept { sz_ = 0; }
00121     void push_back(const T& val) {
00122         if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
00123         data_[sz_++] = val;
00124     }
00125     void push_back(T&& val) {
00126         if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
00127         data_[sz_++] = std::move(val);
00128     }
00129     void pop_back() {
00130         if (sz_ > 0) --sz_;
00131     }
00132     void resize(size_type n, const T& val = T()) {

```

```

00133         if (n > cap_) reserve(n);
00134         if (n > sz_) std::fill(data_ + sz_, data_ + n, val);
00135         sz_ = n;
00136     }
00137     void swap(Vector& other) noexcept {
00138         std::swap(data_, other.data_);
00139         std::swap(sz_, other.sz_);
00140         std::swap(cap_, other.cap_);
00141     }
00142
00143     // Insert element at position
00144     iterator insert(const_iterator pos, const T& value) {
00145         size_type idx = pos - data_;
00146         if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
00147         for (size_type i = sz_; i > idx; --i)
00148             data_[i] = std::move(data_[i - 1]);
00149         data_[idx] = value;
00150         ++sz_;
00151         return data_ + idx;
00152     }
00153
00154     // Erase element at position
00155     iterator erase(const_iterator pos) {
00156         size_type idx = pos - data_;
00157         for (size_type i = idx; i + 1 < sz_; ++i)
00158             data_[i] = std::move(data_[i + 1]);
00159         --sz_;
00160         return data_ + idx;
00161     }
00162
00163     // Assign n copies of value
00164     void assign(size_type n, const T& value) {
00165         if (n > cap_) reserve(n);
00166         std::fill(data_, data_ + n, value);
00167         sz_ = n;
00168     }
00169
00170     // Assign from initializer_list
00171     void assign(std::initializer_list<T> il) {
00172         if (il.size() > cap_) reserve(il.size());
00173         std::copy(il.begin(), il.end(), data_);
00174         sz_ = il.size();
00175     }
00176
00177     // Assign from iterator range
00178     template <typename InputIt>
00179     typename std::enable_if<!std::is_integral<InputIt>::value, void>::type
00180     assign(InputIt first, InputIt last) {
00181         size_type n = std::distance(first, last);
00182         if (n > cap_) reserve(n);
00183         std::copy(first, last, data_);
00184         sz_ = n;
00185     }
00186
00187     // Emplace back
00188     template <typename... Args>
00189     void emplace_back(Args&&... args) {
00190         if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
00191         new (data_ + sz_) T(std::forward<Args>(args)...);
00192         ++sz_;
00193     }
00194
00195     // Insert range [first, last) at position
00196     template <typename InputIt>
00197     iterator insert(const_iterator pos, InputIt first, InputIt last) {
00198         size_type idx = pos - data_;
00199         size_type count = std::distance(first, last);
00200         if (sz_ + count > cap_) reserve(std::max(cap_ * 2, sz_ + count));
00201         for (size_type i = sz_ + count; i-- > idx + count; )
00202             data_[i] = std::move(data_[i - count]);
00203         for (size_type i = 0; i < count; ++i)
00204             data_[idx + i] = *(first++);
00205         sz_ += count;
00206         return data_ + idx;
00207     }
00208
00209     // Insert initializer_list at position
00210     iterator insert(const_iterator pos, std::initializer_list<T> ilist) {
00211         return insert(pos, ilist.begin(), ilist.end());
00212     }
00213
00214     // Erase range [first, last)
00215     iterator erase(const_iterator first, const_iterator last) {
00216         size_type idx_first = first - data_;
00217         size_type idx_last = last - data_;
00218         size_type count = idx_last - idx_first;
00219         for (size_type i = idx_first; i + count < sz_; ++i)

```

```

00220         data_[i] = std::move(data_[i + count]);
00221         sz_ -= count;
00222         return data_ + idx_first;
00223     }
00224
00225     // Emplace element at position
00226     template <typename... Args>
00227     iterator.emplace(const_iterator pos, Args&&... args) {
00228         size_type idx = pos - data_;
00229         if (sz_ == cap_) reserve(cap_ == 0 ? 1 : cap_ * 2);
00230         for (size_type i = sz_; i > idx; --i)
00231             data_[i] = std::move(data_[i - 1]);
00232         new (data_ + idx) T(std::forward<Args>(args)...);
00233         ++sz_;
00234         return data_ + idx;
00235     }
00236
00237     // get_allocator
00238     std::allocator<T> get_allocator() const { return std::allocator<T>(); }
00239
00240     // Comparison operators
00241     bool operator==(const Vector& other) const {
00242         if (sz_ != other.sz_) return false;
00243         for (size_type i = 0; i < sz_; ++i)
00244             if (!data_[i] == other.data_[i]) return false;
00245         return true;
00246     }
00247     bool operator!=(const Vector& other) const { return !(*this == other); }
00248     bool operator<(const Vector& other) const {
00249         return std::lexicographical_compare(begin(), end(), other.begin(), other.end());
00250     }
00251     bool operator>(const Vector& other) const { return other < *this; }
00252     bool operator<=(const Vector& other) const { return !(other < *this); }
00253     bool operator>=(const Vector& other) const { return !(*this < other); }
00254 };
00255
00256 // Non-member swap
00257 template <typename T>
00258 void swap(Vector<T>& a, Vector<T>& b) noexcept {
00259     a.swap(b);
00260 }
00261
00262 #endif // VECTOR_H

```

5.11 Zmogus.h File Reference

```

#include <string>
#include <iostream>

```

Classes

- class [Zmogus](#)

5.12 Zmogus.h

[Go to the documentation of this file.](#)

```

00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003
00004 #include <string>
00005 #include <iostream>
00006
00007 class Zmogus {
00008 protected:
00009     std::string var_;
00010     std::string pav_;
00011 public:
00012     Zmogus() : var_(""), pav_("") {}
00013     Zmogus(const std::string& var, const std::string& pav) : var_(var), pav_(pav) {}

```

```
00014     virtual ~Zmogus() = default;
00015
00016     virtual std::string vardas() const { return var_; }
00017     virtual std::string pavarde() const { return pav_; }
00018
00019     // Grynai virtuali funkcija (abstrakti)
00020     virtual void info() const = 0;
00021 };
00022
00023 #endif // ZMOGUS_H
```

Index

- ~Studentas
 - Studentas, [8](#)
- ~Vector
 - Vector< T >, [13](#)
- ~Zmogus
 - Zmogus, [21](#)
- assign
 - Vector< T >, [14](#)
- at
 - Vector< T >, [14](#)
- back
 - Vector< T >, [14](#)
- begin
 - Vector< T >, [15](#)
- bench.cpp, [23](#)
 - main, [23](#)
- capacity
 - Vector< T >, [15](#)
- clear
 - Vector< T >, [15](#)
- clearInput
 - Studentas.cpp, [26](#)
 - Studentas.h, [28](#)
- const_iterator
 - Vector< T >, [12](#)
- const_pointer
 - Vector< T >, [12](#)
- const_reference
 - Vector< T >, [12](#)
- const_reverse_iterator
 - Vector< T >, [12](#)
- data
 - Vector< T >, [15](#)
- egzaminas
 - Studentas, [8](#)
- emplace
 - Vector< T >, [15](#)
- emplace_back
 - Vector< T >, [15](#)
- empty
 - Vector< T >, [16](#)
- end
 - Vector< T >, [16](#)
- erase
 - Vector< T >, [16](#)
- front
 - Vector< T >, [16](#)
- galutinisMediana
 - Studentas, [8](#)
- galutinisVidurkis
 - Studentas, [9](#)
- generateStudentFiles
 - Studentas.cpp, [26](#)
 - Studentas.h, [28](#)
- generuotiPavarde
 - Studentas.cpp, [26](#)
 - Studentas.h, [29](#)
- generuotiVarda
 - Studentas.cpp, [26](#)
 - Studentas.h, [29](#)
- get_allocator
 - Vector< T >, [16](#)
- handleFileInput
 - Studentas.cpp, [26](#)
 - Studentas.h, [29](#)
- handleOutput
 - Studentas.cpp, [26](#)
 - Studentas.h, [29](#)
- info
 - Studentas, [9](#)
 - Zmogus, [21](#)
- insert
 - Vector< T >, [17](#)
- iterator
 - Vector< T >, [12](#)
- ivestiStudentus
 - Studentas.cpp, [26](#)
 - Studentas.h, [29](#)
- main
 - bench.cpp, [23](#)
 - main.cpp, [24](#)
 - testStudentas.cpp, [31](#)
- main.cpp, [23](#)
 - main, [24](#)
 - MAX_STUDENTU_SKAICIUS, [24](#)
 - promptForSortingMethod, [24](#)
 - rikiavimas, [24](#)
- MAX_STUDENTU_SKAICIUS
 - main.cpp, [24](#)
- Mediana
 - Studentas.cpp, [27](#)

- Studentas.h, 29
- Mylib.h, 24
- namuDarbai
 - Studentas, 9
- operator!=
 - Vector< T >, 17
- operator<
 - Vector< T >, 17
- operator<<
 - Studentas, 10
 - Studentas.cpp, 27
- operator<=
 - Vector< T >, 17
- operator>
 - Vector< T >, 18
- operator>>
 - Studentas, 10
 - Studentas.cpp, 27
- operator>=
 - Vector< T >, 18
- operator=
 - Studentas, 9
 - Vector< T >, 17, 18
- operator==
 - Vector< T >, 18
- operator[]
 - Vector< T >, 18
- pav_
 - Zmogus, 21
- pavarde
 - Studentas, 9
 - Zmogus, 21
- pointer
 - Vector< T >, 12
- pop_back
 - Vector< T >, 18
- promptForSortingMethod
 - main.cpp, 24
- push_back
 - Vector< T >, 18, 19
- rbegin
 - Vector< T >, 19
- reference
 - Vector< T >, 12
- rend
 - Vector< T >, 19
- reserve
 - Vector< T >, 19
- resize
 - Vector< T >, 19
- reverse_iterator
 - Vector< T >, 12
- rikiavimas
 - main.cpp, 24
 - Studentas.cpp, 28
- rikiuotiStudentus
 - Studentas.cpp, 27
 - Studentas.h, 29
- setEgzaminas
 - Studentas, 9
- shrink_to_fit
 - Vector< T >, 19
- size
 - Vector< T >, 20
- size_type
 - Vector< T >, 13
- skaiciuotiCache
 - Studentas, 9
- skaitymas
 - Studentas.cpp, 27
 - Studentas.h, 29
- sortAndOutputStudents
 - Studentas.cpp, 27
 - Studentas.h, 29
- spausdinti
 - Studentas.cpp, 27
 - Studentas.h, 30
- spausdintiStudenta
 - testStudentas.cpp, 31
- Studentas, 7
 - ~Studentas, 8
 - egzaminas, 8
 - galutinisMediana, 8
 - galutinisVidurkis, 9
 - info, 9
 - namuDarbai, 9
 - operator<<, 10
 - operator>>, 10
 - operator=, 9
 - pavarde, 9
 - setEgzaminas, 9
 - skaiciuotiCache, 9
 - Studentas, 8
 - vardas, 9
- Studentas.cpp, 25
 - clearInput, 26
 - generateStudentFiles, 26
 - generuotiPavarde, 26
 - generuotiVarda, 26
 - handleFileInput, 26
 - handleOutput, 26
 - investiStudentus, 26
 - Mediana, 27
 - operator<<, 27
 - operator>>, 27
 - rikiavimas, 28
 - rikiuotiStudentus, 27
 - skaitymas, 27
 - sortAndOutputStudents, 27
 - spausdinti, 27
- Studentas.h, 28
 - clearInput, 28
 - generateStudentFiles, 28

- generuotiPavarde, 29
- generuotiVarda, 29
- handleFileInput, 29
- handleOutput, 29
- ivestiStudentus, 29
- Mediana, 29
- rikiuotiStudentus, 29
- skaitymas, 29
- sortAndOutputStudents, 29
- spausdinti, 30
- swap
 - Vector< T >, 20
 - Vector.h, 32
- testStudentas.cpp, 31
 - main, 31
 - spausdintiStudenta, 31
- value_type
 - Vector< T >, 13
- var_
 - Zmogus, 21
- vardas
 - Studentas, 9
 - Zmogus, 21
- Vector
 - Vector< T >, 13
- Vector< T >, 10
 - ~Vector, 13
 - assign, 14
 - at, 14
 - back, 14
 - begin, 15
 - capacity, 15
 - clear, 15
 - const_iterator, 12
 - const_pointer, 12
 - const_reference, 12
 - const_reverse_iterator, 12
 - data, 15
 - emplace, 15
 - emplace_back, 15
 - empty, 16
 - end, 16
 - erase, 16
 - front, 16
 - get_allocator, 16
 - insert, 17
 - iterator, 12
 - operator!=, 17
 - operator<, 17
 - operator<=, 17
 - operator>, 18
 - operator>=, 18
 - operator=, 17, 18
 - operator==, 18
 - operator[], 18
 - pointer, 12
 - pop_back, 18
 - push_back, 18, 19
 - rbegin, 19
 - reference, 12
 - rend, 19
 - reserve, 19
 - resize, 19
 - reverse_iterator, 12
 - shrink_to_fit, 19
 - size, 20
 - size_type, 13
 - swap, 20
 - value_type, 13
 - Vector, 13
- Vector.h, 31
 - swap, 32
- Zmogus, 20
 - ~Zmogus, 21
 - info, 21
 - pav_, 21
 - pavarde, 21
 - var_, 21
 - vardas, 21
 - Zmogus, 21
- Zmogus.h, 35