

V3

Generated by Doxygen 1.10.0

| | |
|--|----------|
| 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] | 9 |
| 4.1.1.3 ~studentas() | 9 |
| 4.1.1.4 studentas() [3/4] | 9 |
| 4.1.1.5 studentas() [4/4] | 9 |
| 4.1.2 Member Function Documentation | 9 |
| 4.1.2.1 clearEverything() | 9 |
| 4.1.2.2 getEGZ() | 9 |
| 4.1.2.3 getGalutinisM() | 9 |
| 4.1.2.4 getGalutinisV() | 9 |
| 4.1.2.5 getND() | 9 |
| 4.1.2.6 getPavarde() | 10 |
| 4.1.2.7 getVardas() | 10 |
| 4.1.2.8 operator=() [1/2] | 10 |
| 4.1.2.9 operator=() [2/2] | 10 |
| 4.1.2.10 setEGZ() | 10 |
| 4.1.2.11 setND() | 10 |
| 4.1.2.12 setPavarde() | 10 |
| 4.1.2.13 setVardas() | 10 |
| 4.1.3 Friends And Related Symbol Documentation | 11 |
| 4.1.3.1 operator<< | 11 |
| 4.1.3.2 operator>> | 11 |
| 4.2 Vector< T > Class Template Reference | 11 |
| 4.2.1 Member Typedef Documentation | 12 |
| 4.2.1.1 const_iterator | 12 |
| 4.2.1.2 const_reference | 12 |
| 4.2.1.3 difference_type | 12 |
| 4.2.1.4 iterator | 12 |
| 4.2.1.5 reference | 13 |
| 4.2.1.6 size_type | 13 |
| 4.2.1.7 value_type | 13 |

| | |
|--|----|
| 4.2.2 Constructor & Destructor Documentation | 13 |
| 4.2.2.1 Vector() [1/6] | 13 |
| 4.2.2.2 Vector() [2/6] | 13 |
| 4.2.2.3 Vector() [3/6] | 13 |
| 4.2.2.4 Vector() [4/6] | 13 |
| 4.2.2.5 Vector() [5/6] | 13 |
| 4.2.2.6 Vector() [6/6] | 14 |
| 4.2.2.7 ~Vector() | 14 |
| 4.2.3 Member Function Documentation | 14 |
| 4.2.3.1 at() [1/2] | 14 |
| 4.2.3.2 at() [2/2] | 14 |
| 4.2.3.3 back() [1/2] | 14 |
| 4.2.3.4 back() [2/2] | 14 |
| 4.2.3.5 begin() [1/2] | 14 |
| 4.2.3.6 begin() [2/2] | 14 |
| 4.2.3.7 capacity() | 15 |
| 4.2.3.8 cbegin() | 15 |
| 4.2.3.9 cend() | 15 |
| 4.2.3.10 clear() | 15 |
| 4.2.3.11 data() [1/2] | 15 |
| 4.2.3.12 data() [2/2] | 15 |
| 4.2.3.13 empty() | 15 |
| 4.2.3.14 end() [1/2] | 15 |
| 4.2.3.15 end() [2/2] | 15 |
| 4.2.3.16 erase() [1/2] | 16 |
| 4.2.3.17 erase() [2/2] | 16 |
| 4.2.3.18 front() [1/2] | 16 |
| 4.2.3.19 front() [2/2] | 16 |
| 4.2.3.20 insert() [1/2] | 16 |
| 4.2.3.21 insert() [2/2] | 16 |
| 4.2.3.22 max_size() | 16 |
| 4.2.3.23 operator=() [1/3] | 16 |
| 4.2.3.24 operator=() [2/3] | 17 |
| 4.2.3.25 operator=() [3/3] | 17 |
| 4.2.3.26 operator[]() [1/2] | 17 |
| 4.2.3.27 operator[]() [2/2] | 17 |
| 4.2.3.28 pop_back() | 17 |
| 4.2.3.29 push_back() [1/2] | 17 |
| 4.2.3.30 push_back() [2/2] | 17 |
| 4.2.3.31 reserve() | 17 |
| 4.2.3.32 resize() | 18 |
| 4.2.3.33 shrink_to_fit() | 18 |

| | |
|--|-----------|
| 4.2.3.34 size() | 18 |
| 4.2.3.35 swap() | 18 |
| 4.3 zmogus Class Reference | 18 |
| 4.3.1 Constructor & Destructor Documentation | 19 |
| 4.3.1.1 zmogus() [1/2] | 19 |
| 4.3.1.2 zmogus() [2/2] | 19 |
| 4.3.1.3 ~zmogus() | 19 |
| 4.3.2 Member Function Documentation | 19 |
| 4.3.2.1 getPavarde() | 19 |
| 4.3.2.2 getVardas() | 19 |
| 4.3.2.3 setPavarde() | 20 |
| 4.3.2.4 setVardas() | 20 |
| 4.3.3 Member Data Documentation | 20 |
| 4.3.3.1 pavarde | 20 |
| 4.3.3.2 vardas | 20 |
| 5 File Documentation | 21 |
| 5.1 funkcijos.cpp File Reference | 21 |
| 5.1.1 Function Documentation | 22 |
| 5.1.1.1 dis() | 22 |
| 5.1.1.2 dis_lytis() | 22 |
| 5.1.1.3 generuoti() | 22 |
| 5.1.1.4 GeneruotiFailus() | 22 |
| 5.1.1.5 GeneruotiNDPazymius() | 22 |
| 5.1.1.6 GeneruotiVardus() | 22 |
| 5.1.1.7 Ivesti_Pazymius() | 23 |
| 5.1.1.8 Ivesti_Varda() | 23 |
| 5.1.1.9 nd_kiekis() | 23 |
| 5.1.1.10 Netinkamas_Ivestis() | 23 |
| 5.1.1.11 Nuskaityti_Is_Failo() | 23 |
| 5.1.1.12 Rikiuoti_Duomenis() | 23 |
| 5.1.1.13 Skirstyti_Studentus() | 23 |
| 5.1.1.14 Spausdinti_Rezultatus() | 23 |
| 5.1.2 Variable Documentation | 24 |
| 5.1.2.1 lytis | 24 |
| 5.1.2.2 pavardesM | 24 |
| 5.1.2.3 pavardesV | 24 |
| 5.1.2.4 rd | 24 |
| 5.1.2.5 vardaiM | 24 |
| 5.1.2.6 vardaiV | 24 |
| 5.2 funkcijos.h File Reference | 24 |
| 5.2.1 Function Documentation | 25 |

| | |
|----------------------------------|----|
| 5.2.1.1 GeneruotiFailus() | 25 |
| 5.2.1.2 GeneruotiNDPazymius() | 25 |
| 5.2.1.3 GeneruotiVardus() | 26 |
| 5.2.1.4 Ivesti_Pazymius() | 26 |
| 5.2.1.5 Ivesti_Varda() | 26 |
| 5.2.1.6 MedianuRikiavimas() | 26 |
| 5.2.1.7 Netinkamas_Ivestis() | 26 |
| 5.2.1.8 Nuskaityti_Is_Failo() | 26 |
| 5.2.1.9 PavardziuRikiavimas() | 26 |
| 5.2.1.10 Rikiuoti_Duomenis() | 26 |
| 5.2.1.11 Skirstyti_Studentus() | 27 |
| 5.2.1.12 Spausdinti_Rezultatus() | 27 |
| 5.2.1.13 VarduRikiavimas() | 27 |
| 5.2.1.14 VidurkiuRikiavimas() | 27 |
| 5.3 funkcijos.h | 27 |
| 5.4 main.cpp File Reference | 28 |
| 5.4.1 Function Documentation | 28 |
| 5.4.1.1 main() | 28 |
| 5.4.2 Variable Documentation | 28 |
| 5.4.2.1 TaipNe | 28 |
| 5.5 studentai.h File Reference | 29 |
| 5.6 studentai.h | 30 |
| 5.7 vector.h File Reference | 32 |
| 5.7.1 Function Documentation | 34 |
| 5.7.1.1 operator!=(()) | 34 |
| 5.7.1.2 operator<() | 34 |
| 5.7.1.3 operator<=() | 34 |
| 5.7.1.4 operator==(()) | 34 |
| 5.7.1.5 operator>() | 34 |
| 5.7.1.6 operator>=() | 34 |
| 5.7.1.7 swap() | 35 |
| 5.8 vector.h | 35 |

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

| | |
|-------------------------|----|
| Vector< T > | 11 |
| Vector< int > | 11 |
| zmogus | 18 |
| 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 > | 11 |
| zmogus | 18 |

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| | |
|-------------------------------|----|
| funkcijos.cpp | 21 |
| funkcijos.h | 24 |
| main.cpp | 28 |
| studentai.h | 29 |
| vector.h | 32 |

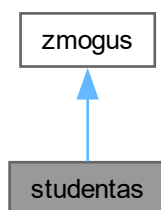
Chapter 4

Class Documentation

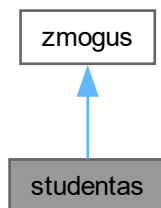
4.1 studentas Class Reference

```
#include <studentai.h>
```

Inheritance diagram for studentas:



Collaboration diagram for studentas:



Public Member Functions

- [studentas](#) ()
- [studentas](#) (const std::string &[vardas](#), const std::string &[pavarde](#), [Vector](#)< int > &ND, int EGZ)
- virtual std::string [getVardas](#) () const override
- virtual std::string [getPavarde](#) () const override
- [~studentas](#) ()
- [studentas](#) (const [studentas](#) &other)
- [studentas](#) ([studentas](#) &&other) noexcept
- [studentas](#) & [operator=](#) (const [studentas](#) &other)
- [studentas](#) & [operator=](#) ([studentas](#) &&other) noexcept
- [Vector](#)< int > [getND](#) () const
- int [getEGZ](#) () const
- double [getGalutinisV](#) () const
- double [getGalutinisM](#) () const
- void [setVardas](#) (const std::string &newName)
- void [setPavarde](#) (const std::string &newSurname)
- void [setND](#) ([Vector](#)< int > &newND)
- void [setEGZ](#) (int newEGZ)
- void [clearEverything](#) ()

Public Member Functions inherited from [zmogus](#)

- [zmogus](#) ()
- [zmogus](#) (const std::string &[vardas](#), const std::string &[pavarde](#))
- [~zmogus](#) ()

Friends

- std::istream & [operator>>](#) (std::istream &is, [studentas](#) &s)
- std::ostream & [operator<<](#) (std::ostream &os, const [studentas](#) &s)

Additional Inherited Members

Protected Attributes inherited from [zmogus](#)

- std::string [vardas](#)
- std::string [pavarde](#)

4.1.1 Constructor & Destructor Documentation

4.1.1.1 [studentas](#)() [1/4]

```
studentas::studentas ( ) [inline]
```

4.1.1.2 studentas() [2/4]

```
studentas::studentas (
    const std::string & vardas,
    const std::string & pavarde,
    Vector< int > & ND,
    int EGZ ) [inline]
```

4.1.1.3 ~studentas()

```
studentas::~~studentas ( ) [inline]
```

4.1.1.4 studentas() [3/4]

```
studentas::studentas (
    const studentas & other ) [inline]
```

4.1.1.5 studentas() [4/4]

```
studentas::studentas (
    studentas && other ) [inline], [noexcept]
```

4.1.2 Member Function Documentation

4.1.2.1 clearEverything()

```
void studentas::clearEverything ( ) [inline]
```

4.1.2.2 getEGZ()

```
int studentas::getEGZ ( ) const [inline]
```

4.1.2.3 getGalutinisM()

```
double studentas::getGalutinisM ( ) const [inline]
```

4.1.2.4 getGalutinisV()

```
double studentas::getGalutinisV ( ) const [inline]
```

4.1.2.5 getND()

```
Vector< int > studentas::getND ( ) const [inline]
```

4.1.2.6 getPavarde()

```
virtual std::string studentas::getPavarde ( ) const [inline], [override], [virtual]
```

Implements [zmogus](#).

4.1.2.7 getVardas()

```
virtual std::string studentas::getVardas ( ) const [inline], [override], [virtual]
```

Implements [zmogus](#).

4.1.2.8 operator=() [1/2]

```
studentas & studentas::operator= (
    const studentas & other ) [inline]
```

4.1.2.9 operator=() [2/2]

```
studentas & studentas::operator= (
    studentas && other ) [inline], [noexcept]
```

4.1.2.10 setEGZ()

```
void studentas::setEGZ (
    int newEGZ ) [inline]
```

4.1.2.11 setND()

```
void studentas::setND (
    Vector< int > & newND ) [inline]
```

4.1.2.12 setPavarde()

```
void studentas::setPavarde (
    const std::string & newSurname ) [inline], [virtual]
```

Reimplemented from [zmogus](#).

4.1.2.13 setVardas()

```
void studentas::setVardas (
    const std::string & newName ) [inline], [virtual]
```

Reimplemented from [zmogus](#).

4.1.3 Friends And Related Symbol Documentation

4.1.3.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const studentas & s ) [friend]
```

4.1.3.2 operator>>

```
std::istream & operator>> (
    std::istream & is,
    studentas & s ) [friend]
```

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

- [studentai.h](#)

4.2 Vector< T > Class Template Reference

```
#include <vector.h>
```

Public Types

- using [value_type](#) = T
- using [iterator](#) = T*
- using [const_iterator](#) = const T*
- using [reference](#) = T&
- using [const_reference](#) = const T&
- using [size_type](#) = std::size_t
- using [difference_type](#) = std::ptrdiff_t

Public Member Functions

- [Vector](#) ()
- [Vector](#) ([size_type](#) count)
- [Vector](#) ([size_type](#) count, const T &value)
- [Vector](#) (std::initializer_list< T > init)
- [Vector](#) (const [Vector](#) &other)
- [Vector](#) ([Vector](#) &&other) noexcept
- [~Vector](#) ()
- [Vector](#) & [operator=](#) (const [Vector](#) &other)
- [Vector](#) & [operator=](#) ([Vector](#) &&other) noexcept
- [Vector](#) & [operator=](#) (std::initializer_list< T > init)
- [reference](#) at ([size_type](#) pos)
- [const_reference](#) at ([size_type](#) pos) const
- [reference](#) [operator\[\]](#) ([size_type](#) pos)
- [const_reference](#) [operator\[\]](#) ([size_type](#) pos) const

- [reference front](#) ()
- [const_reference front](#) () const
- [reference back](#) ()
- [const_reference back](#) () const
- [T * data](#) () noexcept
- [const T * data](#) () const noexcept
- [iterator begin](#) () noexcept
- [const_iterator begin](#) () const noexcept
- [const_iterator cbegin](#) () const noexcept
- [iterator end](#) () noexcept
- [const_iterator end](#) () const noexcept
- [const_iterator cend](#) () const noexcept
- [bool empty](#) () const noexcept
- [size_type size](#) () const noexcept
- [size_type max_size](#) () const noexcept
- [void reserve](#) ([size_type](#) new_cap)
- [size_type capacity](#) () const noexcept
- [void shrink_to_fit](#) ()
- [void clear](#) () noexcept
- [void push_back](#) (const T &value)
- [void push_back](#) (T &&value)
- [void pop_back](#) ()
- [iterator insert](#) ([const_iterator](#) pos, const T &value)
- [iterator insert](#) ([const_iterator](#) pos, T &&value)
- [iterator erase](#) ([const_iterator](#) pos)
- [iterator erase](#) ([const_iterator](#) first, [const_iterator](#) last)
- [void resize](#) ([size_type](#) count, T value=T())
- [void swap](#) ([Vector](#) &other) noexcept

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_reference](#)

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

4.2.1.3 [difference_type](#)

```
template<typename T >
using Vector< T >::difference_type = std::ptrdiff_t
```

4.2.1.4 [iterator](#)

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

4.2.1.5 reference

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

4.2.1.6 size_type

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

4.2.1.7 value_type

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

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Vector() [1/6]

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

4.2.2.2 Vector() [2/6]

```
template<typename T >
Vector< T >::Vector (
    size_type count ) [inline], [explicit]
```

4.2.2.3 Vector() [3/6]

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

4.2.2.4 Vector() [4/6]

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

4.2.2.5 Vector() [5/6]

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

4.2.2.6 Vector() [6/6]

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

4.2.2.7 ~Vector()

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

4.2.3 Member Function Documentation

4.2.3.1 at() [1/2]

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

4.2.3.2 at() [2/2]

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

4.2.3.3 back() [1/2]

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

4.2.3.4 back() [2/2]

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

4.2.3.5 begin() [1/2]

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

4.2.3.6 begin() [2/2]

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

4.2.3.7 capacity()

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

4.2.3.8 cbegin()

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

4.2.3.9 cend()

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

4.2.3.10 clear()

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

4.2.3.11 data() [1/2]

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

4.2.3.12 data() [2/2]

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

4.2.3.13 empty()

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

4.2.3.14 end() [1/2]

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

4.2.3.15 end() [2/2]

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

4.2.3.16 erase() [1/2]

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

4.2.3.17 erase() [2/2]

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

4.2.3.18 front() [1/2]

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

4.2.3.19 front() [2/2]

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

4.2.3.20 insert() [1/2]

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

4.2.3.21 insert() [2/2]

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

4.2.3.22 max_size()

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

4.2.3.23 operator=() [1/3]

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

4.2.3.24 operator=() [2/3]

```
template<typename T >
Vector & Vector< T >::operator= (
    std::initializer_list< T > init ) [inline]
```

4.2.3.25 operator=() [3/3]

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

4.2.3.26 operator[]() [1/2]

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

4.2.3.27 operator[]() [2/2]

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

4.2.3.28 pop_back()

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

4.2.3.29 push_back() [1/2]

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

4.2.3.30 push_back() [2/2]

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

4.2.3.31 reserve()

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

4.2.3.32 `resize()`

```
template<typename T >
void Vector< T >::resize (
    size_type count,
    T value = T() ) [inline]
```

4.2.3.33 `shrink_to_fit()`

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

4.2.3.34 `size()`

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

4.2.3.35 `swap()`

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

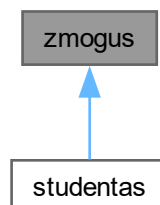
The documentation for this class was generated from the following file:

- [vector.h](#)

4.3 zmogus Class Reference

```
#include <studentai.h>
```

Inheritance diagram for zmogus:



Public Member Functions

- [zmogus](#) ()
- [zmogus](#) (const std::string &[vardas](#), const std::string &[pavarde](#))
- [~zmogus](#) ()
- virtual std::string [getVardas](#) () const =0
- virtual std::string [getPavarde](#) () const =0
- virtual void [setVardas](#) (const std::string &newName)
- virtual void [setPavarde](#) (const std::string &newSurname)

Protected Attributes

- std::string [vardas](#)
- std::string [pavarde](#)

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 & vardas,
    const std::string & pavarde ) [inline]
```

4.3.1.3 [~zmogus\(\)](#)

```
zmogus::~~zmogus ( ) [inline]
```

4.3.2 Member Function Documentation

4.3.2.1 [getPavarde\(\)](#)

```
virtual std::string zmogus::getPavarde ( ) const [pure virtual]
```

Implemented in [studentas](#).

4.3.2.2 [getVardas\(\)](#)

```
virtual std::string zmogus::getVardas ( ) const [pure virtual]
```

Implemented in [studentas](#).

4.3.2.3 setPavarde()

```
virtual void zmogus::setPavarde (
    const std::string & newSurname ) [inline], [virtual]
```

Reimplemented in [studentas](#).

4.3.2.4 setVardas()

```
virtual void zmogus::setVardas (
    const std::string & newName ) [inline], [virtual]
```

Reimplemented in [studentas](#).

4.3.3 Member Data Documentation

4.3.3.1 pavarde

```
std::string zmogus::pavarde [protected]
```

4.3.3.2 vardas

```
std::string zmogus::vardas [protected]
```

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

- [studentai.h](#)

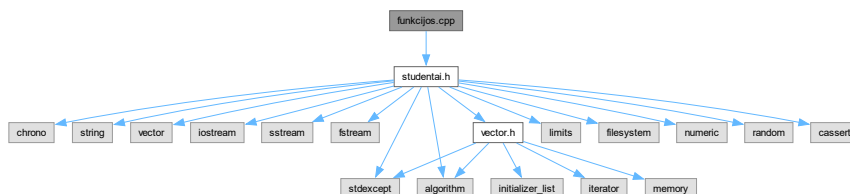
Chapter 5

File Documentation

5.1 funkcijos.cpp File Reference

```
#include "studentai.h"
```

Include dependency graph for funkcijos.cpp:



Functions

- void [Netinkamas_Ivestis](#) (std::string Problema)
- std::mt19937 [generuoti](#) (rd())
- std::uniform_int_distribution< int > [nd_kiekis](#) (5, 20)
- std::uniform_int_distribution< int > [dis](#) (1, 10)
- std::uniform_int_distribution< int > [dis_lytis](#) (0, 1)
- void [GeneruotiNDPazymius](#) (studentas &S, int ND_kiekis)
- void [GeneruotiVardus](#) (studentas &S)
- void [GeneruotiFailus](#) (int reserveDydis, std::string &G_Failo_Vieta)
- void [Ivesti_Pazymius](#) (studentas &S)
- void [Ivesti_Varda](#) (studentas &S)
- std::vector< [studentas](#) > [Nuskaityti_Is_Failo](#) (const std::string &Failo_Pavadinimas, int reserveDydis)
- void [Rikiuoti_Duomenis](#) (std::vector< [studentas](#) > &S)
- void [Skirstyti_Studentus](#) (std::vector< [studentas](#) > &S, std::vector< [studentas](#) > &N, std::vector< [studentas](#) > &G, int Strategija)
- void [Spausdinti_Rezultatus](#) (const std::vector< [studentas](#) > &N, const std::vector< [studentas](#) > &G)

Variables

- `std::random_device` [rd](#)
- `std::vector< std::string >` [vardaiV](#) = { "Jonas", "Petras", "Antanas", "Juozas", "Kazys", "Darius", "Linas", "Tomas", "Giedrius", "Marius" }
- `std::vector< std::string >` [vardaiM](#) = { "Ona", "Maryte", "Aldona", "Gabija", "Dalia", "Danute", "Asta", "Rasa", "Nijole", "Aiste", "Gabriele" }
- `std::vector< std::string >` [pavardesV](#) = { "Jonaitis", "Petraitis", "Antanaitis", "Juozaitis", "Kaziukaitis", "Dariuskaitis", "Linaitis", "Tomaitis", "Giedraitis", "Mariukaitis" }
- `std::vector< std::string >` [pavardesM](#) = { "Jonaite", "Petraityte", "Antanaite", "Juozaitė", "Kaziukaite", "Dariuskaite", "Linaite", "Tomaite", "Giedraite", "Mariukaite", "Antaniene", "Jonaitiene", "Antaniene" }
- `int` [lytis](#) = [dis_lytis\(generuoti\)](#)

5.1.1 Function Documentation

5.1.1.1 `dis()`

```
std::uniform_int_distribution< int > dis (
    1 ,
    10 )
```

5.1.1.2 `dis_lytis()`

```
std::uniform_int_distribution< int > dis_lytis (
    0 ,
    1 )
```

5.1.1.3 `generuoti()`

```
std::mt19937 generuoti (
    rd() )
```

5.1.1.4 `GeneruotiFailus()`

```
void GeneruotiFailus (
    int reserveDydis,
    std::string & G_Failo_Vieta )
```

5.1.1.5 `GeneruotiNDPazymius()`

```
void GeneruotiNDPazymius (
    studentas & S,
    int ND_kiekis )
```

5.1.1.6 `GeneruotiVardus()`

```
void GeneruotiVardus (
    studentas & S )
```

5.1.1.7 Ivesti_Pazymius()

```
void Ivesti_Pazymius (
    studentas & S )
```

5.1.1.8 Ivesti_Varda()

```
void Ivesti_Varda (
    studentas & S )
```

5.1.1.9 nd_kiekis()

```
std::uniform_int_distribution< int > nd_kiekis (
    5 ,
    20 )
```

5.1.1.10 Netinkamas_Ivestis()

```
void Netinkamas_Ivestis (
    std::string Problema )
```

5.1.1.11 Nuskaityti_Is_Failo()

```
std::vector< studentas > Nuskaityti_Is_Failo (
    const std::string & Failo_Pavadinimas,
    int reserveDydis )
```

5.1.1.12 Rikiuoti_Duomenis()

```
void Rikiuoti_Duomenis (
    std::vector< studentas > & S )
```

5.1.1.13 Skirstyti_Studentus()

```
void Skirstyti_Studentus (
    std::vector< studentas > & S,
    std::vector< studentas > & N,
    std::vector< studentas > & G,
    int Strategija )
```

5.1.1.14 Spausdinti_Rezultatus()

```
void Spausdinti_Rezultatus (
    const std::vector< studentas > & N,
    const std::vector< studentas > & G )
```

5.1.2 Variable Documentation

5.1.2.1 lytis

```
int lytis = dis_lytis(generuoti)
```

5.1.2.2 pavardesM

```
std::vector<std::string> pavardesM = { "Jonaite", "Petraityte", "Antanaite", "Juozaitė",
    "Kaziukaite", "Dariukaite", "Linaite", "Tomaite", "Giedraite", "Mariukaite", "Antaniene",
    "Jonaitiene", "Antaniene" }
```

5.1.2.3 pavardesV

```
std::vector<std::string> pavardesV = { "Jonaitis", "Petraitis", "Antanaitis", "Juozaitis",
    "Kaziukaitis", "Dariukaitis", "Linaitis", "Tomaitis", "Giedraitis", "Mariukaitis" }
```

5.1.2.4 rd

```
std::random_device rd
```

5.1.2.5 vardaiM

```
std::vector<std::string> vardaiM = { "Ona", "Maryte", "Aldona", "Gabija", "Dalia", "Danute",
    "Asta", "Rasa", "Nijole", "Aiste", "Gabriele" }
```

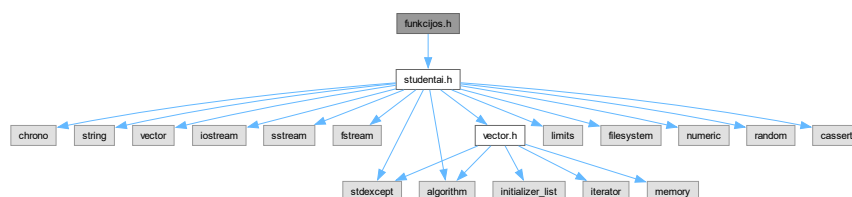
5.1.2.6 vardaiV

```
std::vector<std::string> vardaiV = { "Jonas", "Petras", "Antanas", "Juozas", "Kazys", "Darius",
    "Linas", "Tomas", "Giedrius", "Marius" }
```

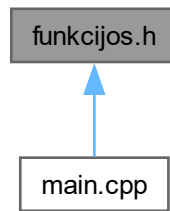
5.2 funkcijos.h File Reference

```
#include "studentai.h"
```

Include dependency graph for funkcijos.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [Netinkamas_Ivestis](#) (std::string Problema)
- void [GeneruotiNDPazymius](#) (studentas &S, int ND_kiekis)
- void [GeneruotiVardus](#) (studentas &S)
- void [GeneruotiFailus](#) (int reserveDydis, std::string &failoPav)
- void [Ivesti_Pazymius](#) (studentas &S)
- void [Ivesti_Varda](#) (studentas &S)
- std::vector< [studentas](#) > [Nuskaityti_Is_Failo](#) (const std::string &Failo_Pavadinimas, int reserveDydis)
- bool [VarduRikiavimas](#) (const [studentas](#) &a, const [studentas](#) &b)
- bool [PavardziuRikiavimas](#) (const [studentas](#) &a, const [studentas](#) &b)
- bool [MedianuRikiavimas](#) (const [studentas](#) &a, const [studentas](#) &b)
- bool [VidurkiuRikiavimas](#) (const [studentas](#) &a, const [studentas](#) &b)
- void [Rikiuoti_Duomenis](#) (std::vector< [studentas](#) > &S)
- void [Skirstyti_Studentus](#) (std::vector< [studentas](#) > &S, std::vector< [studentas](#) > &N, std::vector< [studentas](#) > &G, int Strategija)
- void [Spausdinti_Rezultatus](#) (const std::vector< [studentas](#) > &N, const std::vector< [studentas](#) > &G)

5.2.1 Function Documentation

5.2.1.1 GeneruotiFailus()

```
void GeneruotiFailus (  
    int reserveDydis,  
    std::string & failoPav )
```

5.2.1.2 GeneruotiNDPazymius()

```
void GeneruotiNDPazymius (  
    studentas & S,  
    int ND_kiekis )
```

5.2.1.3 GeneruotiVardus()

```
void GeneruotiVardus (
    studentas & S )
```

5.2.1.4 Ivesti_Pazymius()

```
void Ivesti_Pazymius (
    studentas & S )
```

5.2.1.5 Ivesti_Varda()

```
void Ivesti_Varda (
    studentas & S )
```

5.2.1.6 MedianuRikiavimas()

```
bool MedianuRikiavimas (
    const studentas & a,
    const studentas & b )
```

5.2.1.7 Netinkamas_Ivestis()

```
void Netinkamas_Ivestis (
    std::string Problema )
```

5.2.1.8 Nuskaityti_Is_Failo()

```
std::vector< studentas > Nuskaityti_Is_Failo (
    const std::string & Failo_Pavadinimas,
    int reserveDydis )
```

5.2.1.9 PavardziuRikiavimas()

```
bool PavardziuRikiavimas (
    const studentas & a,
    const studentas & b )
```

5.2.1.10 Rikiuoti_Duomenis()

```
void Rikiuoti_Duomenis (
    std::vector< studentas > & S )
```


5.2.1.11 Skirstyti_Studentus()

```
void Skirstyti_Studentus (
    std::vector< studentas > & S,
    std::vector< studentas > & N,
    std::vector< studentas > & G,
    int Strategija )
```

5.2.1.12 Spausdinti_Rezultatus()

```
void Spausdinti_Rezultatus (
    const std::vector< studentas > & N,
    const std::vector< studentas > & G )
```

5.2.1.13 VarduRikiavimas()

```
bool VarduRikiavimas (
    const studentas & a,
    const studentas & b )
```

5.2.1.14 VidurkiuRikiavimas()

```
bool VidurkiuRikiavimas (
    const studentas & a,
    const studentas & b )
```

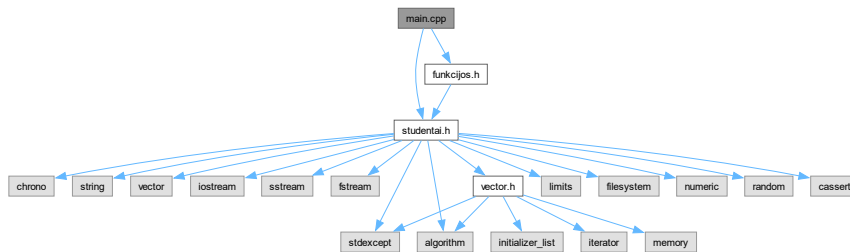
5.3 funkcijos.h

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

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003 #include "studentai.h"
00004
00006 void Netinkamas_Ivestis(std::string Problema);
00007
00009 void GeneruotiNDPazymius(studentas& S, int ND_kiekis);
00010 void GeneruotiVardus(studentas& S);
00011 void GeneruotiFailus(int reserveDydis, std::string& failoPav);
00012
00014 void Ivesti_Pazymius(studentas& S);
00015 void Ivesti_Varda(studentas& S);
00016
00018 std::vector<studentas> Nuskaityti_Is_Failo(const std::string& Failo_Pavadinimas, int reserveDydis);
00019
00021 bool VarduRikiavimas(const studentas& a, const studentas& b);
00022 bool PavardziuRikiavimas(const studentas& a, const studentas& b);
00023 bool MedianuRikiavimas(const studentas& a, const studentas& b);
00024 bool VidurkiuRikiavimas(const studentas& a, const studentas& b);
00025 void Rikiuoti_Duomenis(std::vector<studentas>& S);
00026
00028 void Skirstyti_Studentus(std::vector<studentas>& S, std::vector<studentas>& N, std::vector<studentas>& G, int Strategija);
00029
00031 void Spausdinti_Rezultatus(const std::vector<studentas>& N, const std::vector<studentas>& G);
00032
00033 #endif
```

5.4 main.cpp File Reference

```
#include "studentai.h"  
#include "funkcijos.h"  
Include dependency graph for main.cpp:
```



Functions

- int [main](#) ()

Variables

- char [TaipNe](#)

5.4.1 Function Documentation

5.4.1.1 main()

```
int main ( )
```

5.4.2 Variable Documentation

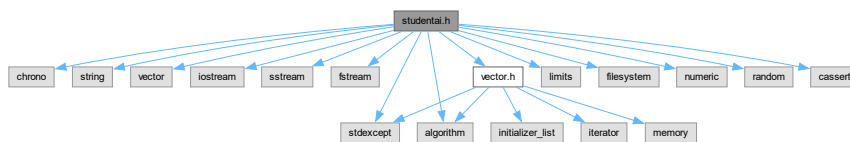
5.4.2.1 TaipNe

```
char TaipNe
```

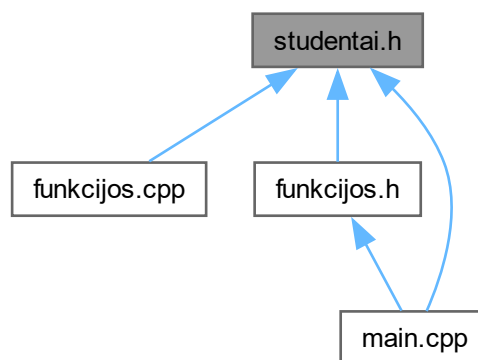
5.5 studentai.h File Reference

```
#include <chrono>
#include <string>
#include <vector>
#include <iostream>
#include <sstream>
#include <fstream>
#include <stdexcept>
#include <limits>
#include <filesystem>
#include <algorithm>
#include <numeric>
#include <random>
#include <cassert>
#include "vector.h"
```

Include dependency graph for studentai.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [zmogus](#)
- class [studentas](#)

5.6 studentai.h

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

```

00001 #ifndef STUDENTAI_H
00002 #define STUDENTAI_H
00003
00004 #include <chrono>
00005 #include <string>
00006 #include <vector>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <stdexcept>
00011 #include <limits>
00012 #include <filesystem>
00013 #include <algorithm>
00014 #include <numeric>
00015 #include <random>
00016 #include <cassert>
00017 #include "vector.h"
00018
00019 //ZMOGUS
00020 class zmogus {
00021 protected:
00022     std::string vardas;
00023     std::string pavarde;
00024 public:
00025     //Konstruktorius
00026     zmogus() : vardas("Bevardis"), pavarde("Bepavardis") { /* std::cout << "Suveike zmogus default
konstruktorius\n"; */ }
00027     zmogus(const std::string& vardas, const std::string& pavarde)
00028         : vardas(vardas), pavarde(pavarde) {}
00029
00030     ~zmogus() {}
00031
00032     virtual std::string getVardas() const = 0;
00033     virtual std::string getPavarde() const = 0;
00034
00035     // Setter'iai
00036     virtual void setVardas(const std::string& newName) { vardas = newName; }
00037     virtual void setPavarde(const std::string& newSurname) { pavarde = newSurname; }
00038
00039 };
00040
00041 //STUDENTAS
00042 class studentas : public zmogus {
00043 private:
00044     Vector<int>ND;
00045     int EGZ;
00046     double GalutinisV;
00047     double GalutinisM;
00048     void ApskaiciuotiGalutinius()
00049     {
00050         if (!ND.empty())
00051         {
00052             GalutinisV = 0.4 * std::accumulate(ND.begin(), ND.end(), 0.0) / ND.size() + 0.6 * EGZ;
00053             if (ND.size() > 1)
00054             {
00055                 Vector<int> sortedND = ND;
00056                 std::sort(sortedND.begin(), sortedND.end());
00057                 size_t mid = sortedND.size() / 2;
00058                 GalutinisM = 0.4 * (sortedND.size() % 2 == 0 ? (sortedND[mid - 1] + sortedND[mid]) /
2.0 : sortedND[mid]) + 0.6 * EGZ;
00059             }
00060             else
00061                 GalutinisM = 0.4 * ND[0] + 0.6 * EGZ;
00062         }
00063     }
00064     else
00065     {
00066         // Jei ND tuščias
00067         GalutinisV = GalutinisM = 0.6 * EGZ;
00068     }
00069 }
00070
00071 public:
00072     studentas() : EGZ(0), ND(), GalutinisV(0), GalutinisM(0) {
00073         //std::cout << "Suveike studentas default konstruktorius\n";
00074     }
00075
00076     studentas(const std::string& vardas, const std::string& pavarde, Vector<int>& ND, int EGZ)
00077         : zmogus(vardas, pavarde), ND(ND), EGZ(EGZ) {
00078         ApskaiciuotiGalutinius();
00079         //std::cout << "Suveike parametrizuotas konstruktorius\n";
00080     }

```

```

00081
00082 // Implementuojame abstrakčius metodus
00083 virtual std::string getVardas() const override {
00084     return vardas;
00085 }
00086
00087 virtual std::string getPavarde() const override {
00088     return pavarde;
00089 }
00090 // Destruktorius
00091 ~studentas() { ND.clear(); /*std::cout << "Suveike destruktoriaus\n";*/ }
00092
00093 // Copy konstruktorius
00094 studentas(const studentas& other)
00095 {
00096     vardas = other.vardas;
00097     pavarde = other.pavarde;
00098     ND = other.ND;
00099     EGZ = other.EGZ;
00100     GalutinisV = other.GalutinisV;
00101     GalutinisM = other.GalutinisM;
00102     //std::cout << "Suveike copy konstruktorius\n";
00103 }
00104 // Move konstruktorius
00105 studentas(studentas&& other) noexcept
00106 {
00107     vardas = std::move(other.vardas);
00108     pavarde = std::move(other.pavarde);
00109     ND = std::move(other.ND);
00110     EGZ = std::move(other.EGZ);
00111     GalutinisV = std::move(other.GalutinisV);
00112     GalutinisM = std::move(other.GalutinisM);
00113     other.clearEverything();
00114     //std::cout << "Suveike move konstruktorius\n";
00115 }
00116 // Copy priskyrimo operatorius
00117 studentas& operator=(const studentas& other)
00118 {
00119
00120     if (this != &other)
00121     {
00122         vardas = other.vardas;
00123         pavarde = other.pavarde;
00124         ND = other.ND;
00125         EGZ = other.EGZ;
00126         GalutinisV = other.GalutinisV;
00127         GalutinisM = other.GalutinisM;
00128         //std::cout << "Suveike copy priskyrimo operatorius\n";
00129     }
00130     return *this;
00131 }
00132 // Move priskyrimo operatorius
00133 studentas& operator=(studentas&& other) noexcept
00134 {
00135
00136     if (this != &other)
00137     {
00138         vardas = std::move(other.vardas);
00139         pavarde = std::move(other.pavarde);
00140         ND = std::move(other.ND);
00141         EGZ = std::move(other.EGZ);
00142         GalutinisV = std::move(other.GalutinisV);
00143         GalutinisM = std::move(other.GalutinisM);
00144         other.clearEverything();
00145         //std::cout << "Suveike move priskyrimo operatorius\n";
00146     }
00147     return *this;
00148 }
00149
00150 // Getter'iai
00151 Vector<int> getND() const { return ND; }
00152 int getEGZ() const { return EGZ; }
00153 double getGalutinisV() const { return GalutinisV; }
00154 double getGalutinisM() const { return GalutinisM; }
00155
00156 // Setter'iai
00157 void setVardas(const std::string& newName) { vardas = newName; }
00158 void setPavarde(const std::string& newSurname) { pavarde = newSurname; }
00159 void setND(Vector<int>& newND) { ND = newND; ApskaiciuotiGalutinius(); }
00160 void setEGZ(int newEGZ) {
00161     EGZ = newEGZ;
00162     ApskaiciuotiGalutinius();
00163 }
00164
00165 friend std::istream& operator>>(std::istream& is, studentas& s)
00166 {
00167     s.vardas.clear();

```

```

00168         s.pavarde.clear();
00169         s.ND.clear();
00170         s.EGZ = 0;
00171
00172
00173         if (!(is » s.vardas » s.pavarde))
00174         {
00175             return is;
00176         }
00177
00178         int pazymys;
00179         Vector<int> NDpazymiai;
00180         while (is » pazymys)
00181         {
00182             NDpazymiai.push_back(pazymys);
00183         }
00184
00185         // Patikrina, ar pasiekė failo pabaigą
00186         if (is.eof()) {
00187             is.clear();
00188         }
00189         // Jei įvedimo operacija nepavyko
00190         else if (is.fail()) {
00191
00192             is.clear();
00193             std::string unused;
00194             std::getline(is, unused);
00195             return is;
00196         }
00197
00198         if (!NDpazymiai.empty())
00199         {
00200             s.EGZ = NDpazymiai.back();
00201             NDpazymiai.pop_back();
00202             s.ND = NDpazymiai;
00203         }
00204
00205         s.ApskaiciuotiGalutinius();
00206         //std::cout « "Suveike ivesties operatorius\n";
00207         return is;
00208     }
00209
00210     friend std::ostream& operator<(std::ostream& os, const studentas& s)
00211     {
00212         os « std::setw(20) « s.pavarde « std::setw(20) « s.vardas « std::setw(20) «
00213         std::setprecision(3) « s.GalutinisV « std::setw(20) « std::setprecision(3) « s.GalutinisM « std::endl;
00214         //std::cout « "Suveike ivesties operatorius\n";
00215         return os;
00216     }
00217
00218     void clearEverything()
00219     {
00220         this->vardas.clear();
00221         this->pavarde.clear();
00222         this->ND.clear();
00223         this->EGZ = 0;
00224         this->GalutinisV = 0;
00225         this->GalutinisM = 0;
00226     }
00227 };
00228 #endif

```

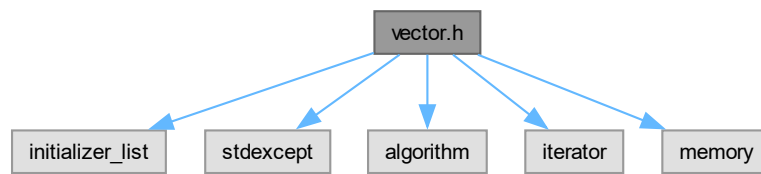
5.7 vector.h File Reference

```

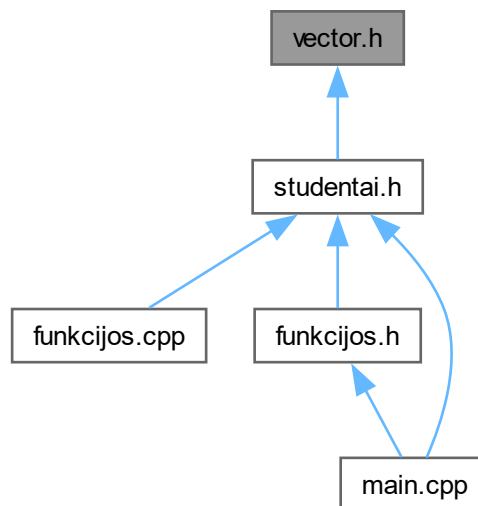
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <memory>

```

Include dependency graph for vector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `Vector< T >`

Functions

- `template<typename T >`
`bool operator== (const Vector< T > &lhs, const Vector< T > &rhs)`
- `template<typename T >`
`bool operator!= (const Vector< T > &lhs, const Vector< T > &rhs)`
- `template<typename T >`
`bool operator< (const Vector< T > &lhs, const Vector< T > &rhs)`
- `template<typename T >`
`bool operator<= (const Vector< T > &lhs, const Vector< T > &rhs)`

- `template<typename T >`
`bool operator> (const Vector< T > &lhs, const Vector< T > &rhs)`
- `template<typename T >`
`bool operator>= (const Vector< T > &lhs, const Vector< T > &rhs)`
- `template<typename T >`
`void swap (Vector< T > &lhs, Vector< T > &rhs) noexcept`

5.7.1 Function Documentation

5.7.1.1 operator!=(())

```
template<typename T >
bool operator!= (
    const Vector< T > & lhs,
    const Vector< T > & rhs )
```

5.7.1.2 operator<()

```
template<typename T >
bool operator< (
    const Vector< T > & lhs,
    const Vector< T > & rhs )
```

5.7.1.3 operator<=()

```
template<typename T >
bool operator<= (
    const Vector< T > & lhs,
    const Vector< T > & rhs )
```

5.7.1.4 operator==()

```
template<typename T >
bool operator== (
    const Vector< T > & lhs,
    const Vector< T > & rhs )
```

5.7.1.5 operator>()

```
template<typename T >
bool operator> (
    const Vector< T > & lhs,
    const Vector< T > & rhs )
```

5.7.1.6 operator>=()

```
template<typename T >
bool operator>= (
    const Vector< T > & lhs,
    const Vector< T > & rhs )
```


5.7.1.7 swap()

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

5.8 vector.h

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

```
00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <initializer_list>
00005 #include <stdexcept>
00006 #include <algorithm>
00007 #include <iterator>
00008 #include <memory>
00009
00010 template <typename T>
00011 class Vector {
00012 public:
00013     // Member types
00014     using value_type = T;
00015     using iterator = T*;
00016     using const_iterator = const T*;
00017     using reference = T&;
00018     using const_reference = const T&;
00019     using size_type = std::size_t;
00020     using difference_type = std::ptrdiff_t;
00021
00022     // Constructors
00023     Vector() : data_(nullptr), size_(0), capacity_(0) {}
00024
00025     explicit Vector(size_type count) {
00026         data_ = new T[count]();
00027         size_ = count;
00028         capacity_ = count;
00029     }
00030
00031     Vector(size_type count, const T& value) {
00032         data_ = new T[count];
00033         size_ = count;
00034         capacity_ = count;
00035         std::fill_n(data_, count, value);
00036     }
00037
00038     Vector(std::initializer_list<T> init) : Vector(init.size()) {
00039         std::copy(init.begin(), init.end(), data_);
00040     }
00041
00042     Vector(const Vector& other) : Vector(other.size_) {
00043         std::copy(other.data_, other.data_ + other.size_, data_);
00044     }
00045
00046     Vector(Vector&& other) noexcept : data_(other.data_), size_(other.size_),
00047         capacity_(other.capacity_) {
00048         other.data_ = nullptr;
00049         other.size_ = 0;
00050         other.capacity_ = 0;
00051     }
00052
00053     // Destructor
00054     ~Vector() {
00055         delete[] data_;
00056     }
00057
00058     // Assignment operators
00059     Vector& operator=(const Vector& other) {
00060         if (this != &other) {
00061             Vector temp(other);
00062             swap(temp);
00063         }
00064         return *this;
00065     }
00066
00067     Vector& operator=(Vector&& other) noexcept {
00068         if (this != &other) {
```

```

00068         delete[] data_;
00069         data_ = other.data_;
00070         size_ = other.size_;
00071         capacity_ = other.capacity_;
00072         other.data_ = nullptr;
00073         other.size_ = 0;
00074         other.capacity_ = 0;
00075     }
00076     return *this;
00077 }
00078
00079 Vector& operator=(std::initializer_list<T> init) {
00080     Vector temp(init);
00081     swap(temp);
00082     return *this;
00083 }
00084
00085 // Element access
00086 reference at(size_type pos) {
00087     if (pos >= size_) throw std::out_of_range("Vector::at");
00088     return data_[pos];
00089 }
00090
00091 const_reference at(size_type pos) const {
00092     if (pos >= size_) throw std::out_of_range("Vector::at");
00093     return data_[pos];
00094 }
00095
00096 reference operator[](size_type pos) {
00097     return data_[pos];
00098 }
00099
00100 const_reference operator[](size_type pos) const {
00101     return data_[pos];
00102 }
00103
00104 reference front() {
00105     return data_[0];
00106 }
00107
00108 const_reference front() const {
00109     return data_[0];
00110 }
00111
00112 reference back() {
00113     return data_[size_ - 1];
00114 }
00115
00116 const_reference back() const {
00117     return data_[size_ - 1];
00118 }
00119
00120 T* data() noexcept {
00121     return data_;
00122 }
00123
00124 const T* data() const noexcept {
00125     return data_;
00126 }
00127
00128 // Iterators
00129 iterator begin() noexcept {
00130     return data_;
00131 }
00132
00133 const_iterator begin() const noexcept {
00134     return data_;
00135 }
00136
00137 const_iterator cbegin() const noexcept {
00138     return data_;
00139 }
00140
00141 iterator end() noexcept {
00142     return data_ + size_;
00143 }
00144
00145 const_iterator end() const noexcept {
00146     return data_ + size_;
00147 }
00148
00149 const_iterator cend() const noexcept {
00150     return data_ + size_;
00151 }
00152
00153 // Capacity
00154 bool empty() const noexcept {

```

```

00155         return size_ == 0;
00156     }
00157
00158     size_type size() const noexcept {
00159         return size_;
00160     }
00161
00162     size_type max_size() const noexcept {
00163         return std::numeric_limits<size_type>::max();
00164     }
00165
00166     void reserve(size_type new_cap) {
00167         if (new_cap > capacity_) {
00168             T* new_data = new T[new_cap];
00169             std::copy(data_, data_ + size_, new_data);
00170             delete[] data_;
00171             data_ = new_data;
00172             capacity_ = new_cap;
00173         }
00174     }
00175
00176     size_type capacity() const noexcept {
00177         return capacity_;
00178     }
00179
00180     void shrink_to_fit() {
00181         if (capacity_ > size_) {
00182             T* new_data = new T[size_];
00183             std::copy(data_, data_ + size_, new_data);
00184             delete[] data_;
00185             data_ = new_data;
00186             capacity_ = size_;
00187         }
00188     }
00189
00190     // Modifiers
00191     void clear() noexcept {
00192         size_ = 0;
00193     }
00194
00195     void push_back(const T& value) {
00196         if (size_ == capacity_) reserve(capacity_ > 0 ? 2 * capacity_ : 1);
00197         data_[size_++] = value;
00198     }
00199
00200     void push_back(T&& value) {
00201         if (size_ == capacity_) reserve(capacity_ > 0 ? 2 * capacity_ : 1);
00202         data_[size_++] = std::move(value);
00203     }
00204
00205     void pop_back() {
00206         if (size_ > 0) --size_;
00207     }
00208
00209     iterator insert(const_iterator pos, const T& value) {
00210         size_type index = pos - data_;
00211         if (size_ == capacity_) reserve(capacity_ > 0 ? 2 * capacity_ : 1);
00212         for (size_type i = size_; i > index; --i) {
00213             data_[i] = data_[i - 1];
00214         }
00215         data_[index] = value;
00216         ++size_;
00217         return data_ + index;
00218     }
00219
00220     iterator insert(const_iterator pos, T&& value) {
00221         size_type index = pos - data_;
00222         if (size_ == capacity_) reserve(capacity_ > 0 ? 2 * capacity_ : 1);
00223         for (size_type i = size_; i > index; --i) {
00224             data_[i] = data_[i - 1];
00225         }
00226         data_[index] = std::move(value);
00227         ++size_;
00228         return data_ + index;
00229     }
00230
00231     iterator erase(const_iterator pos) {
00232         size_type index = pos - data_;
00233         for (size_type i = index; i < size_ - 1; ++i) {
00234             data_[i] = std::move(data_[i + 1]);
00235         }
00236         --size_;
00237         return data_ + index;
00238     }
00239
00240     iterator erase(const_iterator first, const_iterator last) {
00241         size_type start_index = first - data_;

```

```

00242         size_type end_index = last - data_;
00243         size_type count = end_index - start_index;
00244         for (size_type i = start_index; i < size_ - count; ++i) {
00245             data_[i] = std::move(data_[i + count]);
00246         }
00247         size_ -= count;
00248         return data_ + start_index;
00249     }
00250
00251     void resize(size_type count, T value = T()) {
00252         if (count > size_) {
00253             reserve(count);
00254             std::fill(data_ + size_, data_ + count, value);
00255         }
00256         size_ = count;
00257     }
00258
00259     void swap(Vector& other) noexcept {
00260         std::swap(data_, other.data_);
00261         std::swap(size_, other.size_);
00262         std::swap(capacity_, other.capacity_);
00263     }
00264
00265 private:
00266     T* data_;
00267     size_type size_;
00268     size_type capacity_;
00269 };
00270
00271 // Non-member functions
00272 template <typename T>
00273 bool operator==(const Vector<T>& lhs, const Vector<T>& rhs) {
00274     return lhs.size() == rhs.size() && std::equal(lhs.begin(), lhs.end(), rhs.begin());
00275 }
00276
00277 template <typename T>
00278 bool operator!=(const Vector<T>& lhs, const Vector<T>& rhs) {
00279     return !(lhs == rhs);
00280 }
00281
00282 template <typename T>
00283 bool operator<(const Vector<T>& lhs, const Vector<T>& rhs) {
00284     return std::lexicographical_compare(lhs.begin(), lhs.end(), rhs.begin(), rhs.end());
00285 }
00286
00287 template <typename T>
00288 bool operator<=(const Vector<T>& lhs, const Vector<T>& rhs) {
00289     return !(rhs < lhs);
00290 }
00291
00292 template <typename T>
00293 bool operator>(const Vector<T>& lhs, const Vector<T>& rhs) {
00294     return rhs < lhs;
00295 }
00296
00297 template <typename T>
00298 bool operator>=(const Vector<T>& lhs, const Vector<T>& rhs) {
00299     return !(lhs < rhs);
00300 }
00301
00302 template <typename T>
00303 void swap(Vector<T>& lhs, Vector<T>& rhs) noexcept {
00304     lhs.swap(rhs);
00305 }
00306
00307 #endif

```

Index

~Vector
 Vector< T >, [14](#)
~studentas
 studentas, [9](#)
~zmogus
 zmogus, [19](#)

at
 Vector< T >, [14](#)

back
 Vector< T >, [14](#)
begin
 Vector< T >, [14](#)

capacity
 Vector< T >, [14](#)
cbegin
 Vector< T >, [15](#)
cend
 Vector< T >, [15](#)
clear
 Vector< T >, [15](#)
clearEverything
 studentas, [9](#)
const_iterator
 Vector< T >, [12](#)
const_reference
 Vector< T >, [12](#)

data
 Vector< T >, [15](#)
difference_type
 Vector< T >, [12](#)
dis
 funkcijos.cpp, [22](#)
dis_lytis
 funkcijos.cpp, [22](#)

empty
 Vector< T >, [15](#)
end
 Vector< T >, [15](#)
erase
 Vector< T >, [15](#), [16](#)

front
 Vector< T >, [16](#)
funkcijos.cpp, [21](#)
 dis, [22](#)
 dis_lytis, [22](#)

generuoti, [22](#)
GeneruotiFailus, [22](#)
GeneruotiNDPazymius, [22](#)
GeneruotiVardus, [22](#)
Ivesti_Pazymius, [22](#)
Ivesti_Varda, [23](#)
lytis, [24](#)
nd_kiekis, [23](#)
Netinkamas_Ivestis, [23](#)
Nuskaityti_Is_Failo, [23](#)
pavardesM, [24](#)
pavardesV, [24](#)
rd, [24](#)
Rikiuoti_Duomenis, [23](#)
Skirstyti_Studentus, [23](#)
Spausdinti_Rezultatus, [23](#)
vardaiM, [24](#)
vardaiV, [24](#)
funkcijos.h, [24](#)
 GeneruotiFailus, [25](#)
 GeneruotiNDPazymius, [25](#)
 GeneruotiVardus, [25](#)
 Ivesti_Pazymius, [26](#)
 Ivesti_Varda, [26](#)
 MedianuRikiavimas, [26](#)
 Netinkamas_Ivestis, [26](#)
 Nuskaityti_Is_Failo, [26](#)
 PavardziuRikiavimas, [26](#)
 Rikiuoti_Duomenis, [26](#)
 Skirstyti_Studentus, [26](#)
 Spausdinti_Rezultatus, [27](#)
 VarduRikiavimas, [27](#)
 VidurkiuRikiavimas, [27](#)

generuoti
 funkcijos.cpp, [22](#)
GeneruotiFailus
 funkcijos.cpp, [22](#)
 funkcijos.h, [25](#)
GeneruotiNDPazymius
 funkcijos.cpp, [22](#)
 funkcijos.h, [25](#)
GeneruotiVardus
 funkcijos.cpp, [22](#)
 funkcijos.h, [25](#)
getEGZ
 studentas, [9](#)
getGalutinisM
 studentas, [9](#)
getGalutinisV

- studentas, 9
- getND
 - studentas, 9
- getPavarde
 - studentas, 9
 - zmogus, 19
- getVarDas
 - studentas, 10
 - zmogus, 19
- insert
 - Vector< T >, 16
- iterator
 - Vector< T >, 12
- lvesti_Pazymius
 - funkcijos.cpp, 22
 - funkcijos.h, 26
- lvesti_Varda
 - funkcijos.cpp, 23
 - funkcijos.h, 26
- lytis
 - funkcijos.cpp, 24
- main
 - main.cpp, 28
- main.cpp, 28
 - main, 28
 - TaipNe, 28
- max_size
 - Vector< T >, 16
- MedianuRikiavimas
 - funkcijos.h, 26
- nd_kiekis
 - funkcijos.cpp, 23
- Netinkamas_lvestis
 - funkcijos.cpp, 23
 - funkcijos.h, 26
- Nuskaityti_Is_Failo
 - funkcijos.cpp, 23
 - funkcijos.h, 26
- operator!=
 - vector.h, 34
- operator<
 - vector.h, 34
- operator<<
 - studentas, 11
- operator<=
 - vector.h, 34
- operator>
 - vector.h, 34
- operator>>
 - studentas, 11
- operator>=
 - vector.h, 34
- operator=
 - studentas, 10
- Vector< T >, 16, 17
- operator==
 - vector.h, 34
- operator[]
 - Vector< T >, 17
- pavarde
 - zmogus, 20
- pavardesM
 - funkcijos.cpp, 24
- pavardesV
 - funkcijos.cpp, 24
- PavardziuRikiavimas
 - funkcijos.h, 26
- pop_back
 - Vector< T >, 17
- push_back
 - Vector< T >, 17
- rd
 - funkcijos.cpp, 24
- reference
 - Vector< T >, 12
- reserve
 - Vector< T >, 17
- resize
 - Vector< T >, 17
- Rikiuoti_Duomenis
 - funkcijos.cpp, 23
 - funkcijos.h, 26
- setEGZ
 - studentas, 10
- setND
 - studentas, 10
- setPavarde
 - studentas, 10
 - zmogus, 19
- setVardas
 - studentas, 10
 - zmogus, 20
- shrink_to_fit
 - Vector< T >, 18
- size
 - Vector< T >, 18
- size_type
 - Vector< T >, 13
- Skirstyti_Studentus
 - funkcijos.cpp, 23
 - funkcijos.h, 26
- Spausdinti_Rezultatus
 - funkcijos.cpp, 23
 - funkcijos.h, 27
- studentai.h, 29
- studentas, 7
 - ~studentas, 9
 - clearEverything, 9
 - getEGZ, 9
 - getGalutinisM, 9

- getGalutinisV, 9
- getND, 9
- getPavarde, 9
- getVardas, 10
- operator<<, 11
- operator>>, 11
- operator=, 10
- setEGZ, 10
- setND, 10
- setPavarde, 10
- setVardas, 10
- studentas, 8, 9
- swap
 - Vector< T >, 18
 - vector.h, 34
- TaipNe
 - main.cpp, 28
- value_type
 - Vector< T >, 13
- vardaiM
 - funkcijos.cpp, 24
- vardaiV
 - funkcijos.cpp, 24
- vardas
 - zmogus, 20
- VarduRikiavimas
 - funkcijos.h, 27
- Vector
 - Vector< T >, 13
- Vector< T >, 11
 - ~Vector, 14
 - at, 14
 - back, 14
 - begin, 14
 - capacity, 14
 - cbegin, 15
 - cend, 15
 - clear, 15
 - const_iterator, 12
 - const_reference, 12
 - data, 15
 - difference_type, 12
 - empty, 15
 - end, 15
 - erase, 15, 16
 - front, 16
 - insert, 16
 - iterator, 12
 - max_size, 16
 - operator=, 16, 17
 - operator[], 17
 - pop_back, 17
 - push_back, 17
 - reference, 12
 - reserve, 17
 - resize, 17
 - shrink_to_fit, 18
 - size, 18
 - size_type, 13
 - swap, 18
 - value_type, 13
 - Vector, 13
- vector.h, 32
 - operator!=, 34
 - operator<, 34
 - operator<=, 34
 - operator>, 34
 - operator>=, 34
 - operator==, 34
 - swap, 34
- VidurkiuRikiavimas
 - funkcijos.h, 27
- zmogus, 18
 - ~zmogus, 19
 - getPavarde, 19
 - getVardas, 19
 - pavarde, 20
 - setPavarde, 19
 - setVardas, 20
 - vardas, 20
 - zmogus, 19