

3.0 projektas

Generated by Doxygen 1.10.0



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 studentas Struct Reference	7
4.1.1 Member Data Documentation	7
4.1.1.1 balas	7
4.1.1.2 egz	8
4.1.1.3 mediana	8
4.1.1.4 nd	8
4.1.1.5 pavarde	8
4.1.1.6 vardas	8
4.1.1.7 vid	8
4.2 Studentas Class Reference	8
4.2.1 Constructor & Destructor Documentation	10
4.2.1.1 Studentas() [1/4]	10
4.2.1.2 Studentas() [2/4]	10
4.2.1.3 ~Studentas()	10
4.2.1.4 Studentas() [3/4]	10
4.2.1.5 Studentas() [4/4]	10
4.2.2 Member Function Documentation	10
4.2.2.1 getBalas()	10
4.2.2.2 getEgz()	11
4.2.2.3 getMediana()	11
4.2.2.4 getNd()	11
4.2.2.5 getPavarde()	11
4.2.2.6 getVardas()	11
4.2.2.7 getVid()	11
4.2.2.8 operator=() [1/2]	11
4.2.2.9 operator=() [2/2]	11
4.2.2.10 setBalas()	11
4.2.2.11 setEgz()	12
4.2.2.12 setMediana()	12
4.2.2.13 setNd()	12
4.2.2.14 setPavarde()	12
4.2.2.15 setVardas()	12
4.2.2.16 setVid()	12

4.2.3 Friends And Related Symbol Documentation	12
4.2.3.1 operator<<	12
4.2.3.2 operator>>	13
4.3 Vector< T > Class Template Reference	13
4.3.1 Member Typedef Documentation	14
4.3.1.1 const_iterator	14
4.3.1.2 const_reference	15
4.3.1.3 iterator	15
4.3.1.4 reference	15
4.3.1.5 size_type	15
4.3.1.6 value_type	15
4.3.2 Constructor & Destructor Documentation	15
4.3.2.1 Vector() [1/6]	15
4.3.2.2 Vector() [2/6]	15
4.3.2.3 Vector() [3/6]	16
4.3.2.4 Vector() [4/6]	16
4.3.2.5 Vector() [5/6]	16
4.3.2.6 Vector() [6/6]	16
4.3.2.7 ~Vector()	16
4.3.3 Member Function Documentation	16
4.3.3.1 assign() [1/3]	16
4.3.3.2 assign() [2/3]	17
4.3.3.3 assign() [3/3]	17
4.3.3.4 at() [1/2]	17
4.3.3.5 at() [2/2]	17
4.3.3.6 back() [1/2]	17
4.3.3.7 back() [2/2]	17
4.3.3.8 begin() [1/2]	17
4.3.3.9 begin() [2/2]	18
4.3.3.10 capacity()	18
4.3.3.11 clear()	18
4.3.3.12 data() [1/2]	18
4.3.3.13 data() [2/2]	18
4.3.3.14 empty()	18
4.3.3.15 end() [1/2]	18
4.3.3.16 end() [2/2]	18
4.3.3.17 erase() [1/2]	19
4.3.3.18 erase() [2/2]	19
4.3.3.19 front() [1/2]	19
4.3.3.20 front() [2/2]	19
4.3.3.21 insert() [1/2]	19
4.3.3.22 insert() [2/2]	19

4.3.3.23 max_size()	19
4.3.3.24 operator std::vector< T >()	20
4.3.3.25 operator"!=()	20
4.3.3.26 operator<()	20
4.3.3.27 operator<=()	20
4.3.3.28 operator=() [1/2]	20
4.3.3.29 operator=() [2/2]	20
4.3.3.30 operator==()	20
4.3.3.31 operator>()	21
4.3.3.32 operator>=()	21
4.3.3.33 operator[]() [1/2]	21
4.3.3.34 operator[]() [2/2]	21
4.3.3.35 pop_back()	21
4.3.3.36 push_back() [1/2]	21
4.3.3.37 push_back() [2/2]	21
4.3.3.38 reserve()	21
4.3.3.39 resize() [1/2]	22
4.3.3.40 resize() [2/2]	22
4.3.3.41 shrink_to_fit()	22
4.3.3.42 size()	22
4.3.3.43 swap() [1/2]	22
4.3.3.44 swap() [2/2]	22
4.4 Zmogus Class Reference	23
4.4.1 Constructor & Destructor Documentation	23
4.4.1.1 Zmogus() [1/2]	23
4.4.1.2 Zmogus() [2/2]	23
4.4.1.3 ~Zmogus()	24
4.4.2 Member Function Documentation	24
4.4.2.1 getPavarde()	24
4.4.2.2 getVardas()	24
4.4.2.3 setPavarde()	24
4.4.2.4 setVardas()	24
4.4.3 Member Data Documentation	24
4.4.3.1 pavarde_	24
4.4.3.2 vardas_	24
<b>5 File Documentation</b>	<b>25</b>
5.1 fun.cpp File Reference	25
5.1.1 Function Documentation	26
5.1.1.1 generavimasFailo()	26
5.1.1.2 generavimasPazymiu()	26
5.1.1.3 generavimasPazymiuCase2()	26

5.1.1.4 generavimasStudentu()	26
5.1.1.5 isValidName()	26
5.1.1.6 padalintiStudentus()	26
5.1.1.7 pagalMediana()	26
5.1.1.8 pagalPavarde()	26
5.1.1.9 pagalVarda()	27
5.1.1.10 pagalVidurki()	27
5.1.1.11 skaiciavimas()	27
5.1.1.12 skaitymas()	27
5.1.1.13 skaitymasTeksto()	27
5.1.1.14 spausdinti()	27
5.1.1.15 spausdintiTeksto()	27
5.1.1.16 test_constructor()	27
5.1.1.17 test_copy_assignment()	27
5.1.1.18 test_copy_constructor()	28
5.1.1.19 test_input_operator()	28
5.1.1.20 test_move_assignment()	28
5.1.1.21 test_move_constructor()	28
5.1.1.22 test_output_operator()	28
5.2 fun.h File Reference	28
5.2.1 Function Documentation	29
5.2.1.1 generavimasFailo()	29
5.2.1.2 generavimasPazymiu()	29
5.2.1.3 generavimasPazymiuCase2()	29
5.2.1.4 generavimasStudentu()	30
5.2.1.5 isValidName()	30
5.2.1.6 padalintiStudentus()	30
5.2.1.7 pagalMediana()	30
5.2.1.8 pagalPavarde()	30
5.2.1.9 pagalVarda()	30
5.2.1.10 pagalVidurki()	30
5.2.1.11 skaiciavimas()	30
5.2.1.12 skaitymas()	31
5.2.1.13 skaitymasTeksto()	31
5.2.1.14 spausdinti()	31
5.2.1.15 spausdintiTeksto()	31
5.2.2 Variable Documentation	31
5.2.2.1 MAX_ND_SIZE	31
5.2.2.2 MAX_STUDENTS	31
5.3 fun.h	31
5.4 main.cpp File Reference	32
5.4.1 Function Documentation	32

5.4.1.1 main()	32
5.5 studentas.h File Reference	33
5.5.1 Function Documentation	34
5.5.1.1 generavimasFailo()	34
5.5.1.2 generavimasPazymiu()	34
5.5.1.3 generavimasPazymiuCase2()	34
5.5.1.4 generavimasStudentu()	34
5.5.1.5 isValidName()	35
5.5.1.6 padalintiStudentus()	35
5.5.1.7 pagalMediana()	35
5.5.1.8 pagalPavarde()	35
5.5.1.9 pagalVarda()	35
5.5.1.10 pagalVidurki()	35
5.5.1.11 skaiciavimas()	35
5.5.1.12 skaitymas()	35
5.5.1.13 skaitymasTeksto()	36
5.5.1.14 spausdinti()	36
5.5.1.15 spausdintiTeksto()	36
5.5.1.16 test_constructor()	36
5.5.1.17 test_copy_assignment()	36
5.5.1.18 test_copy_constructor()	36
5.5.1.19 test_input_operator()	36
5.5.1.20 test_move_assignment()	36
5.5.1.21 test_move_constructor()	36
5.5.1.22 test_output_operator()	36
5.5.2 Variable Documentation	37
5.5.2.1 MAX_ND_SIZE	37
5.5.2.2 MAX_STUDENTS	37
5.6 studentas.h	37
5.7 test.cpp File Reference	39
5.7.1 Function Documentation	40
5.7.1.1 TEST() [1/5]	40
5.7.1.2 TEST() [2/5]	40
5.7.1.3 TEST() [3/5]	40
5.7.1.4 TEST() [4/5]	40
5.7.1.5 TEST() [5/5]	40
5.8 vector.h File Reference	40
5.9 vector.h	41
<b>Index</b>	<b>45</b>





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

studentas . . . . .	7
Vector< T > . . . . .	13
Vector< int > . . . . .	13
Zmogus . . . . .	23
Studentas . . . . .	8



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">studentas</a>	7
<a href="#">Studentas</a>	8
<a href="#">Vector&lt; T &gt;</a>	13
<a href="#">Zmogus</a>	23



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">fun.cpp</a>	25
<a href="#">fun.h</a>	28
<a href="#">main.cpp</a>	32
<a href="#">studentas.h</a>	33
<a href="#">test.cpp</a>	39
<a href="#">vector.h</a>	40



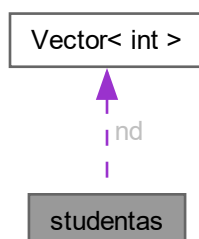
## Chapter 4

# Class Documentation

### 4.1 studentas Struct Reference

```
#include <fun.h>
```

Collaboration diagram for studentas:



#### Public Attributes

- `string vardas`
- `string pavarde`
- `Vector< int > nd`
- `int egz`
- `double balas`
- `double vid`
- `double mediana`

#### 4.1.1 Member Data Documentation

##### 4.1.1.1 balas

```
double studentas::balas
```

#### 4.1.1.2 egz

```
int studentas::egz
```

#### 4.1.1.3 mediana

```
double studentas::mediana
```

#### 4.1.1.4 nd

```
Vector<int> studentas::nd
```

#### 4.1.1.5 pavarde

```
string studentas::pavarde
```

#### 4.1.1.6 vardas

```
string studentas::vardas
```

#### 4.1.1.7 vid

```
double studentas::vid
```

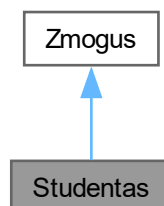
The documentation for this struct was generated from the following file:

- [fun.h](#)

## 4.2 Studentas Class Reference

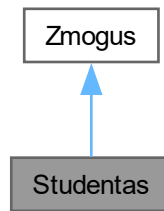
```
#include <studentas.h>
```

Inheritance diagram for Studentas:





Collaboration diagram for Studentas:



### Public Member Functions

- `Studentas ()`
- `Studentas (const string &vardas, const string &pavarde, const Vector< int > &nd, int egz, double balas, double vid, double mediana)`
- `~Studentas ()`
- `Studentas (const Studentas &other)`
- `Studentas & operator= (const Studentas &other)`
- `Studentas (Studentas &&other) noexcept`
- `Studentas & operator= (Studentas &&other) noexcept`
- `string getVardas () const override`
- `string getPavarde () const override`
- `Vector< int > & getNd ()`
- `int getEgz () const`
- `double getBalas () const`
- `double getVid () const`
- `double getMediana () const`
- `void setVardas (const string &vardas)`
- `void setPavarde (const string &pavarde)`
- `void setNd (const Vector< int > &nd)`
- `void setEgz (const int &egz)`
- `void setBalas (const double &balas)`
- `void setVid (const double &vid)`
- `void setMediana (const double &mediana)`

### Public Member Functions inherited from **Zmogus**

- `Zmogus ()=default`
- `Zmogus (const string &vardas, const string &pavarde)`
- `virtual ~Zmogus ()`

### Friends

- `istream & operator>> (istream &is, Studentas &student)`
- `ostream & operator<< (ostream &os, const Studentas &student)`

## Additional Inherited Members

### Protected Attributes inherited from [Zmogus](#)

- [string vardas\\_](#)
- [string pavarde\\_](#)

## 4.2.1 Constructor & Destructor Documentation

### 4.2.1.1 Studentas() [1/4]

```
Studentas::Studentas ( ) [inline]
```

### 4.2.1.2 Studentas() [2/4]

```
Studentas::Studentas (
    const string & vardas,
    const string & pavarde,
    const Vector< int > & nd,
    int egz,
    double balas,
    double vid,
    double mediana ) [inline]
```

### 4.2.1.3 ~Studentas()

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

### 4.2.1.4 Studentas() [3/4]

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

### 4.2.1.5 Studentas() [4/4]

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

## 4.2.2 Member Function Documentation

### 4.2.2.1 getBalas()

```
double Studentas::getBalas ( ) const [inline]
```

#### 4.2.2.2 getEgz()

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

#### 4.2.2.3 getMediana()

```
double Studentas::getMediana ( ) const [inline]
```

#### 4.2.2.4 getNd()

```
Vector< int > & Studentas::getNd ( ) [inline]
```

#### 4.2.2.5 getPavarde()

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

Implements [Zmogus](#).

#### 4.2.2.6 getVardas()

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

Implements [Zmogus](#).

#### 4.2.2.7 getVid()

```
double Studentas::getVid ( ) const [inline]
```

#### 4.2.2.8 operator=() [1/2]

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

#### 4.2.2.9 operator=() [2/2]

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

#### 4.2.2.10 setBalas()

```
void Studentas::setBalas (
    const double & balas ) [inline]
```

#### 4.2.2.11 setEgz()

```
void Studentas::setEgz (
    const int & egz ) [inline]
```

#### 4.2.2.12 setMediana()

```
void Studentas::setMediana (
    const double & mediana ) [inline]
```

#### 4.2.2.13 setNd()

```
void Studentas::setNd (
    const Vector< int > & nd ) [inline]
```

#### 4.2.2.14 setPavarde()

```
void Studentas::setPavarde (
    const string & pavarde ) [inline], [virtual]
```

Reimplemented from [Zmogus](#).

#### 4.2.2.15 setVardas()

```
void Studentas::setVardas (
    const string & vardas ) [inline], [virtual]
```

Reimplemented from [Zmogus](#).

#### 4.2.2.16 setVid()

```
void Studentas::setVid (
    const double & vid ) [inline]
```

### 4.2.3 Friends And Related Symbol Documentation

#### 4.2.3.1 operator<<

```
ostream & operator<< (
    ostream & os,
    const Studentas & student ) [friend]
```

## 4.2.3.2 operator&gt;&gt;

```
istream & operator>> (
    istream & is,
    Studentas & student ) [friend]
```

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

- [studentas.h](#)

## 4.3 Vector&lt; T &gt; Class Template Reference

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

## Public Types

- [typedef T value\\_type](#)  
*MEMBER TYPES.*
- [typedef size\\_t size\\_type](#)
- [typedef T & reference](#)
- [typedef const T & const\\_reference](#)
- [typedef T \\* iterator](#)
- [typedef const T \\* const\\_iterator](#)

## Public Member Functions

- [Vector \(\)](#)
- [Vector \(size\\_type n, const T &t=T{}\)](#)  
*fill*
- [Vector \(const Vector &v\)](#)  
*copy konstruktorius*
- [template<class InputIterator > Vector \(InputIterator first, InputIterator last\)](#)  
*range konstruktorius*
- [Vector \(Vector &&v\)](#)  
*move konstruktorius*
- [Vector \(const std::initializer\\_list< T > il\)](#)  
*initializer list konstruktorius*
- [~Vector \(\)](#)  
*destruktorius*
- [operator std::vector< T > \(\) const](#)  
*operator=*
- [Vector & operator= \(const Vector &other\)](#)  
*copy assignment operatorius*
- [Vector & operator= \(Vector &&other\)](#)  
*move assignment operatorius*
- [template<class InputIterator > void assign \(InputIterator first, InputIterator last\)](#)  
*assign*

- `void assign (size_type n, const value_type &val)`
- `void assign (std::initializer_list< value_type > il)`
- `const_reference at (size_type n) const`
- element access*
- `T & operator[] (size_type n)`
- `const T & operator[] (size_type n) const`
- `reference at (size_type n)`
- `reference front ()`
- `const_reference front () const`
- `reference back ()`
- `const_reference back () const`
- `value_type * data () noexcept`
- `const value_type * data () const noexcept`
- `iterator begin ()`
- iterators*
- `const_iterator begin () const`
- `iterator end ()`
- `const_iterator end () const`
- `size_type size () const`
- capacity*
- `size_type max_size () const`
- `void resize (size_type sz)`
- `void resize (size_type sz, const value_type &value)`
- `size_type capacity () const`
- `bool empty () const noexcept`
- `void reserve (size_type n)`
- `void shrink_to_fit ()`
- `void clear () noexcept`
- modifiers*
- `iterator insert (const_iterator position, const value_type &val)`
- `iterator insert (iterator position, size_type n, const value_type &val)`
- `iterator erase (iterator position)`
- `iterator erase (iterator first, iterator last)`
- `void push_back (const value_type &t)`
- `void push_back (value_type &&val)`
- `void pop_back ()`
- `void swap (Vector &x)`
- `bool operator== (const Vector< T > &other) const`
- NON-MEMBER FUNCTIONS.*
- `bool operator!= (const Vector< T > &other) const`
- `bool operator< (const Vector< T > &other) const`
- `bool operator<= (const Vector< T > &other) const`
- `bool operator> (const Vector< T > &other) const`
- `bool operator>= (const Vector< T > &other) const`
- `void swap (Vector< T > &x, Vector< T > &y)`

## 4.3.1 Member Typedef Documentation

### 4.3.1.1 `const_iterator`

```
template<typename T >
typedef const T* Vector< T >::const_iterator
```

#### 4.3.1.2 const\_reference

```
template<typename T >
typedef const T& Vector< T >::const_reference
```

#### 4.3.1.3 iterator

```
template<typename T >
typedef T* Vector< T >::iterator
```

#### 4.3.1.4 reference

```
template<typename T >
typedef T& Vector< T >::reference
```

#### 4.3.1.5 size\_type

```
template<typename T >
typedef size_t Vector< T >::size_type
```

#### 4.3.1.6 value\_type

```
template<typename T >
typedef T Vector< T >::value_type
```

MEMBER TYPES.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 Vector() [1/6]

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

MEMBER FUNCTIONS default konstruktorius

#### 4.3.2.2 Vector() [2/6]

```
template<typename T >
Vector< T >::Vector (
    size_type n,
    const T & t = T{} ) [inline], [explicit]
```

fill

**4.3.2.3 Vector()** [3/6]

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

copy konstruktorius

**4.3.2.4 Vector()** [4/6]

```
template<typename T >
template<class InputIterator >
Vector< T >::Vector (
    InputIterator first,
    InputIterator last ) [inline]
```

range konstruktorius

**4.3.2.5 Vector()** [5/6]

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

move konstruktorius

**4.3.2.6 Vector()** [6/6]

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

initializer list konstruktorius

**4.3.2.7 ~Vector()**

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

destruktorius

**4.3.3 Member Function Documentation****4.3.3.1 assign()** [1/3]

```
template<typename T >
template<class InputIterator >
void Vector< T >::assign (
    InputIterator first,
    InputIterator last ) [inline]
```

assign



**4.3.3.2 assign()** [2/3]

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

**4.3.3.3 assign()** [3/3]

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

**4.3.3.4 at()** [1/2]

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

**4.3.3.5 at()** [2/2]

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

element access

**4.3.3.6 back()** [1/2]

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

**4.3.3.7 back()** [2/2]

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

**4.3.3.8 begin()** [1/2]

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

iterators

#### 4.3.3.9 begin() [2/2]

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

#### 4.3.3.10 capacity()

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

#### 4.3.3.11 clear()

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

modifiers

#### 4.3.3.12 data() [1/2]

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

#### 4.3.3.13 data() [2/2]

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

#### 4.3.3.14 empty()

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

#### 4.3.3.15 end() [1/2]

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

#### 4.3.3.16 end() [2/2]

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

**4.3.3.17 erase()** [1/2]

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

**4.3.3.18 erase()** [2/2]

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

**4.3.3.19 front()** [1/2]

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

**4.3.3.20 front()** [2/2]

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

**4.3.3.21 insert()** [1/2]

```
template<typename T >
iterator Vector< T >::insert (
    const_iterator position,
    const value_type & val ) [inline]
```

**4.3.3.22 insert()** [2/2]

```
template<typename T >
iterator Vector< T >::insert (
    iterator position,
    size_type n,
    const value_type & val ) [inline]
```

**4.3.3.23 max\_size()**

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

**4.3.3.24 operator std::vector< T >()**

```
template<typename T >
Vector< T >::operator std::vector< T > ( ) const [inline]

operator=
```

**4.3.3.25 operator"!="()**

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

**4.3.3.26 operator<()**

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

**4.3.3.27 operator<=()**

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

**4.3.3.28 operator=() [1/2]**

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

copy assignment operatorius

**4.3.3.29 operator=() [2/2]**

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

move assignment operatorius

**4.3.3.30 operator==( )**

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

NON-MEMBER FUNCTIONS.

**4.3.3.31 operator>()**

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

**4.3.3.32 operator>=()**

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

**4.3.3.33 operator[]() [1/2]**

```
template<typename T >
T & Vector< T >::operator[] (
    size_type n ) [inline]
```

**4.3.3.34 operator[]() [2/2]**

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

**4.3.3.35 pop\_back()**

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

**4.3.3.36 push\_back() [1/2]**

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

**4.3.3.37 push\_back() [2/2]**

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

**4.3.3.38 reserve()**

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

**4.3.3.39** `resize()` [1/2]

```
template<typename T >
void Vector< T >::resize (
    size_type sz ) [inline]
```

**4.3.3.40** `resize()` [2/2]

```
template<typename T >
void Vector< T >::resize (
    size_type sz,
    const value_type & value ) [inline]
```

**4.3.3.41** `shrink_to_fit()`

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

**4.3.3.42** `size()`

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

capacity

**4.3.3.43** `swap()` [1/2]

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

**4.3.3.44** `swap()` [2/2]

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

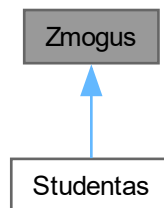
The documentation for this class was generated from the following file:

- [vector.h](#)

## 4.4 Zmogus Class Reference

```
#include <studentas.h>
```

Inheritance diagram for Zmogus:



### Public Member Functions

- [Zmogus \(\)](#)=default
- [Zmogus \(const string &vardas, const string &pavarde\)](#)
- [virtual ~Zmogus \(\)](#)
- [virtual void setVardas \(const string &vardas\)](#)
- [virtual void setPavarde \(const string &pavarde\)](#)
- [virtual string getVardas \(\) const](#) =0
- [virtual string getPavarde \(\) const](#) =0

### Protected Attributes

- [string vardas\\_](#)
- [string pavarde\\_](#)

### 4.4.1 Constructor & Destructor Documentation

#### 4.4.1.1 Zmogus() [1/2]

```
Zmogus::Zmogus ( ) [default]
```

#### 4.4.1.2 Zmogus() [2/2]

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

#### 4.4.1.3 ~Zmogus()

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

### 4.4.2 Member Function Documentation

#### 4.4.2.1 getPavarde()

```
virtual string Zmogus::getPavarde ( ) const [pure virtual]
```

Implemented in [Studentas](#).

#### 4.4.2.2 getVardas()

```
virtual string Zmogus::getVardas ( ) const [pure virtual]
```

Implemented in [Studentas](#).

#### 4.4.2.3 setPavarde()

```
virtual void Zmogus::setPavarde (
    const string & pavarde ) [inline], [virtual]
```

Reimplemented in [Studentas](#).

#### 4.4.2.4 setVardas()

```
virtual void Zmogus::setVardas (
    const string & vardas ) [inline], [virtual]
```

Reimplemented in [Studentas](#).

### 4.4.3 Member Data Documentation

#### 4.4.3.1 pavarde\_

```
string Zmogus::pavarde_ [protected]
```

#### 4.4.3.2 vardas\_

```
string Zmogus::vardas_ [protected]
```

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

- [studentas.h](#)

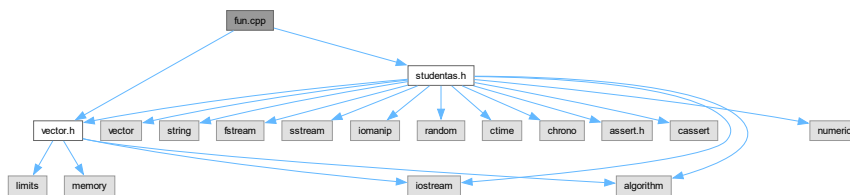


# Chapter 5

## File Documentation

### 5.1 fun.cpp File Reference

```
#include "studentas.h"  
#include "vector.h"  
Include dependency graph for fun.cpp:
```



#### Functions

- `bool isValidName (const string &name)`
- `bool pagalVarda (const Studentas &A, const Studentas &B)`
- `bool pagalPavarde (const Studentas &A, const Studentas &B)`
- `bool pagalVidurki (const Studentas &A, const Studentas &B)`
- `bool pagalMediana (const Studentas &A, const Studentas &B)`
- `void skaitymas (Vector< Studentas > &A, int n)`
- `void skaitymasTeksto (Vector< Studentas > &A)`
- `void skaiciavimas (Vector< Studentas > &A)`
- `void padalintiStudentus (Vector< Studentas > &A)`
- `void spausdinti (const Vector< Studentas > &A)`
- `void spausdintiTeksto (const Vector< Studentas > &A)`
- `void generavimasPazymiu (Vector< Studentas > &A, int n2)`
- `void generavimasPazymiuCase2 (Vector< Studentas > &A)`
- `void generavimasStudentu (Vector< Studentas > &A, int n)`
- `void generavimasFailo (int kiekis)`
- `void test_constructor ()`
- `void test_copy_constructor ()`
- `void test_move_constructor ()`
- `void test_copy_assignment ()`
- `void test_move_assignment ()`
- `void test_input_operator ()`
- `void test_output_operator ()`

## 5.1.1 Function Documentation

### 5.1.1.1 generavimasFailo()

```
void generavimasFailo (
    int kiekis )
```

### 5.1.1.2 generavimasPazymiu()

```
void generavimasPazymiu (
    Vector< Studentas > & A,
    int n2 )
```

### 5.1.1.3 generavimasPazymiuCase2()

```
void generavimasPazymiuCase2 (
    Vector< Studentas > & A )
```

### 5.1.1.4 generavimasStudentu()

```
void generavimasStudentu (
    Vector< Studentas > & A,
    int n )
```

### 5.1.1.5 isValidName()

```
bool isValidName (
    const string & name )
```

### 5.1.1.6 padalintiStudentus()

```
void padalintiStudentus (
    Vector< Studentas > & A )
```

### 5.1.1.7 pagalMediana()

```
bool pagalMediana (
    const Studentas & A,
    const Studentas & B )
```

### 5.1.1.8 pagalPavarde()

```
bool pagalPavarde (
    const Studentas & A,
    const Studentas & B )
```

#### 5.1.1.9 pagalVarda()

```
bool pagalVarda (
    const Studentas & A,
    const Studentas & B )
```

#### 5.1.1.10 pagalVidurki()

```
bool pagalVidurki (
    const Studentas & A,
    const Studentas & B )
```

#### 5.1.1.11 skaiciavimas()

```
void skaiciavimas (
    Vector< Studentas > & A )
```

#### 5.1.1.12 skaitymas()

```
void skaitymas (
    Vector< Studentas > & A,
    int n )
```

#### 5.1.1.13 skaitymasTeksto()

```
void skaitymasTeksto (
    Vector< Studentas > & A )
```

#### 5.1.1.14 spausdinti()

```
void spausdinti (
    const Vector< Studentas > & A )
```

#### 5.1.1.15 spausdintiTeksto()

```
void spausdintiTeksto (
    const Vector< Studentas > & A )
```

#### 5.1.1.16 test\_constructor()

```
void test_constructor ( )
```

#### 5.1.1.17 test\_copy\_assignment()

```
void test_copy_assignment ( )
```

#### 5.1.1.18 test\_copy\_constructor()

```
void test_copy_constructor ( )
```

#### 5.1.1.19 test\_input\_operator()

```
void test_input_operator ( )
```

#### 5.1.1.20 test\_move\_assignment()

```
void test_move_assignment ( )
```

#### 5.1.1.21 test\_move\_constructor()

```
void test_move_constructor ( )
```

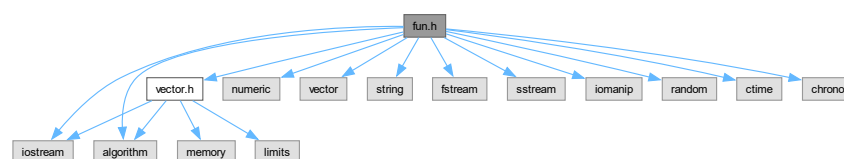
#### 5.1.1.22 test\_output\_operator()

```
void test_output_operator ( )
```

## 5.2 fun.h File Reference

```
#include <iostream>
#include <numeric>
#include <vector>
#include <string>
#include <algorithm>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <random>
#include <ctime>
#include <chrono>
#include "vector.h"
```

Include dependency graph for fun.h:



## Classes

- struct [studentas](#)

## Functions

- void [skaitymas](#) (Vector< [studentas](#) > &A, int n)
- void [skaitymasTeksto](#) (Vector< [studentas](#) > &A)
- bool [pagalVarda](#) (const [studentas](#) &A, const [studentas](#) &B)
- bool [pagalPavarde](#) (const [studentas](#) &A, const [studentas](#) &B)
- bool [pagalVidurki](#) (const [studentas](#) &A, const [studentas](#) &B)
- bool [pagalMediana](#) (const [studentas](#) &A, const [studentas](#) &B)
- void [spausdinti](#) (const Vector< [studentas](#) > &A)
- void [spausdintiTeksto](#) (const Vector< [studentas](#) > &A)
- void [skaiciavimas](#) (Vector< [studentas](#) > &A)
- void [padalintiStudentus](#) (Vector< [studentas](#) > &A)
- void [generavimasPazymiu](#) (Vector< [studentas](#) > &A, int n)
- void [generavimasPazymiuCase2](#) (Vector< [studentas](#) > &A)
- void [generavimasStudentu](#) (Vector< [studentas](#) > &A, int n)
- void [generavimasFailo](#) (Vector< [studentas](#) > &A, int n)
- bool [isValidName](#) (const string &name)

## Variables

- const int [MAX\\_ND\\_SIZE](#) = 500
- const int [MAX\\_STUDENTS](#) = 500

## 5.2.1 Function Documentation

### 5.2.1.1 [generavimasFailo\(\)](#)

```
void generavimasFailo (
    Vector< studentas > & A,
    int n )
```

### 5.2.1.2 [generavimasPazymiu\(\)](#)

```
void generavimasPazymiu (
    Vector< studentas > & A,
    int n )
```

### 5.2.1.3 [generavimasPazymiuCase2\(\)](#)

```
void generavimasPazymiuCase2 (
    Vector< studentas > & A )
```

#### 5.2.1.4 generavimasStudentu()

```
void generavimasStudentu (
    Vector< studentas > & A,
    int n )
```

#### 5.2.1.5 isValidName()

```
bool isValidName (
    const string & name )
```

#### 5.2.1.6 padalintiStudentus()

```
void padalintiStudentus (
    Vector< studentas > & A )
```

#### 5.2.1.7 pagalMediana()

```
bool pagalMediana (
    const studentas & A,
    const studentas & B )
```

#### 5.2.1.8 pagalPavarde()

```
bool pagalPavarde (
    const studentas & A,
    const studentas & B )
```

#### 5.2.1.9 pagalVarda()

```
bool pagalVarda (
    const studentas & A,
    const studentas & B )
```

#### 5.2.1.10 pagalVidurki()

```
bool pagalVidurki (
    const studentas & A,
    const studentas & B )
```

#### 5.2.1.11 skaiciavimas()

```
void skaiciavimas (
    Vector< studentas > & A )
```

#### 5.2.1.12 skaitymas()

```
void skaitymas (
    Vector< studentas > & A,
    int n )
```

#### 5.2.1.13 skaitymasTeksto()

```
void skaitymasTeksto (
    Vector< studentas > & A )
```

#### 5.2.1.14 spausdinti()

```
void spausdinti (
    const Vector< studentas > & A )
```

#### 5.2.1.15 spausdintiTeksto()

```
void spausdintiTeksto (
    const Vector< studentas > & A )
```

### 5.2.2 Variable Documentation

#### 5.2.2.1 MAX\_ND\_SIZE

```
const int MAX_ND_SIZE = 500
```

#### 5.2.2.2 MAX\_STUDENTS

```
const int MAX_STUDENTS = 500
```

## 5.3 fun.h

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

```
00001 #ifndef LABAS
00002 #define LABAS
00003 #include <iostream>
00004 #include <numeric>
00005 #include <vector>
00006 #include <string>
00007 #include <algorithm>
00008 #include <fstream>
00009 #include <sstream>
00010 #include <iomanip>
00011 #include <random>
00012 #include <ctime>
00013 #include <chrono>
00014 #include "vector.h"
00015 using namespace std;
00016
00017 const int MAX_ND_SIZE = 500;
00018 const int MAX_STUDENTS = 500;
```

```

00019
00020 struct studentas
00021 {
00022     string vardas;
00023     string pavarde;
00024     Vector <int> nd;
00025     int egz;
00026     double balas;
00027     double vid;
00028     double mediana;
00029 };
00030
00031 void skaitymas (Vector <studentas> & A, int n);
00032 void skaitymasTeksto (Vector <studentas> & A);
00033 bool pagalVarda(const studentas & A, const studentas &B);
00034 bool pagalPavarde(const studentas &A, const studentas &B);
00035 bool pagalVidurki(const studentas &A, const studentas &B);
00036 bool pagalMediana(const studentas &A, const studentas &B);
00037 void spausdinti (const Vector <studentas> & A);
00038 void spausdintiTeksto(const Vector <studentas> & A);
00039 void skaiciavimas (Vector <studentas> & A);
00040 void padalintiStudentus (Vector <studentas> &A);
00041 void generavimasPazymiu (Vector <studentas> & A, int n);
00042 void generavimasPazymiuCase2 (Vector <studentas> & A);
00043 void generavimasStudentu (Vector <studentas> & A, int n);
00044 void generavimasFailo (Vector <studentas> & A, int n);
00045 bool isValidName(const string &name);
00046 #endif

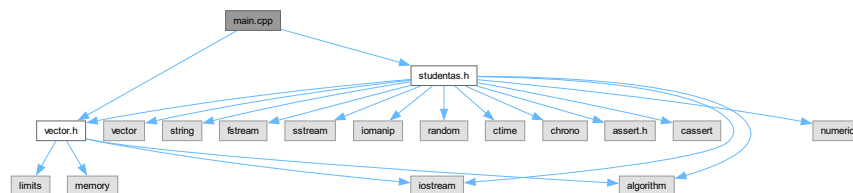
```

## 5.4 main.cpp File Reference

```
#include "studentas.h"
```

```
#include "vector.h"
```

Include dependency graph for main.cpp:



### Functions

- `int main ()`

#### 5.4.1 Function Documentation

##### 5.4.1.1 main()

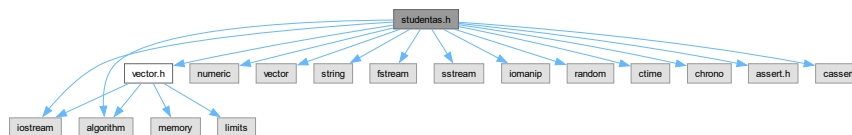
```
int main ( )
```



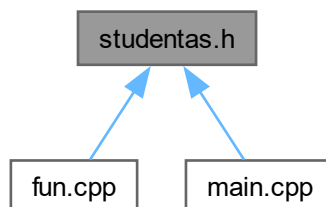
## 5.5 studentas.h File Reference

```
#include <iostream>
#include <numeric>
#include <vector>
#include <string>
#include <algorithm>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <random>
#include <ctime>
#include <chrono>
#include <assert.h>
#include <cassert>
#include "vector.h"
```

Include dependency graph for studentas.h:



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



### Classes

- class [Zmogus](#)
- class [Studentas](#)

### Functions

- void [skaitymas](#) ([Vector](#)< [Studentas](#) > &, int)
- void [skaitymasTeksto](#) ([Vector](#)< [Studentas](#) > &)
- bool [pagalVarda](#) (const [Studentas](#) &, const [Studentas](#) &)

- `bool pagalPavarde (const Studentas &, const Studentas &)`
- `bool pagalVidurki (const Studentas &, const Studentas &)`
- `bool pagalMediana (const Studentas &, const Studentas &)`
- `void spausdinti (const Vector< Studentas > &)`
- `void spausdintiTeksto (const Vector< Studentas > &)`
- `void skaiciavimas (Vector< Studentas > &)`
- `void padalintiStudentus (Vector< Studentas > &)`
- `void generavimasPazymiu (Vector< Studentas > &, int n)`
- `void generavimasPazymiuCase2 (Vector< Studentas > &)`
- `void generavimasStudentu (Vector< Studentas > &, int)`
- `void generavimasFailo (Vector< Studentas > &, int)`
- `bool isValidName (const string &)`
- `void test_constructor ()`
- `void test_copy_constructor ()`
- `void test_move_constructor ()`
- `void test_copy_assignment ()`
- `void test_move_assignment ()`
- `void test_input_operator ()`
- `void test_output_operator ()`

## Variables

- `const int MAX_ND_SIZE = 500`
- `const int MAX_STUDENTS = 500`

## 5.5.1 Function Documentation

### 5.5.1.1 generavimasFailo()

```
void generavimasFailo (
    Vector< Studentas > & ,
    int )
```

### 5.5.1.2 generavimasPazymiu()

```
void generavimasPazymiu (
    Vector< Studentas > & A,
    int n )
```

### 5.5.1.3 generavimasPazymiuCase2()

```
void generavimasPazymiuCase2 (
    Vector< Studentas > & A )
```

### 5.5.1.4 generavimasStudentu()

```
void generavimasStudentu (
    Vector< Studentas > & A,
    int n )
```

#### 5.5.1.5 isValidName()

```
bool isValidName (
    const string & name )
```

#### 5.5.1.6 padalintiStudentus()

```
void padalintiStudentus (
    Vector< Studentas > & A )
```

#### 5.5.1.7 pagalMediana()

```
bool pagalMediana (
    const Studentas & A,
    const Studentas & B )
```

#### 5.5.1.8 pagalPavarde()

```
bool pagalPavarde (
    const Studentas & A,
    const Studentas & B )
```

#### 5.5.1.9 pagalVarda()

```
bool pagalVarda (
    const Studentas & A,
    const Studentas & B )
```

#### 5.5.1.10 pagalVidurki()

```
bool pagalVidurki (
    const Studentas & A,
    const Studentas & B )
```

#### 5.5.1.11 skaiciavimas()

```
void skaiciavimas (
    Vector< Studentas > & A )
```

#### 5.5.1.12 skaitymas()

```
void skaitymas (
    Vector< Studentas > & A,
    int n )
```

#### 5.5.1.13 skaitymasTeksto()

```
void skaitymasTeksto (
    Vector< Studentas > & A )
```

#### 5.5.1.14 spausdinti()

```
void spausdinti (
    const Vector< Studentas > & A )
```

#### 5.5.1.15 spausdintiTeksto()

```
void spausdintiTeksto (
    const Vector< Studentas > & A )
```

#### 5.5.1.16 test\_constructor()

```
void test_constructor ( )
```

#### 5.5.1.17 test\_copy\_assignment()

```
void test_copy_assignment ( )
```

#### 5.5.1.18 test\_copy\_constructor()

```
void test_copy_constructor ( )
```

#### 5.5.1.19 test\_input\_operator()

```
void test_input_operator ( )
```

#### 5.5.1.20 test\_move\_assignment()

```
void test_move_assignment ( )
```

#### 5.5.1.21 test\_move\_constructor()

```
void test_move_constructor ( )
```

#### 5.5.1.22 test\_output\_operator()

```
void test_output_operator ( )
```

## 5.5.2 Variable Documentation

### 5.5.2.1 MAX\_ND\_SIZE

```
const int MAX_ND_SIZE = 500
```

### 5.5.2.2 MAX\_STUDENTS

```
const int MAX_STUDENTS = 500
```

## 5.6 studentas.h

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

```
00001 #ifndef KA
00002 #define KA
00003
00004 #include <iostream>
00005 #include <numeric>
00006 #include <vector>
00007 #include <string>
00008 #include <algorithm>
00009 #include <fstream>
00010 #include <sstream>
00011 #include <iomanip>
00012 #include <random>
00013 #include <ctime>
00014 #include <chrono>
00015 #include <assert.h>
00016 #include <cassert>
00017 #include "vector.h"
00018 using namespace std;
00019
00020 const int MAX_ND_SIZE = 500;
00021 const int MAX_STUDENTS = 500;
00022
00023 class Zmogus {
00024     protected:
00025         string vardas_;
00026         string pavarde_;
00027     public:
00028         Zmogus() = default;
00029         Zmogus(const string& vardas, const string& pavarde) : vardas_(vardas), pavarde_(pavarde) {}
00030         virtual ~Zmogus() {}
00031
00032         virtual void setVardas(const string& vardas) { vardas_ = vardas; }
00033         virtual void setPavarde(const string& pavarde) { pavarde_ = pavarde; }
00034         virtual string getVardas() const = 0; // { return vardas_; }
00035         virtual string getPavarde() const = 0; // { return pavarde_; }
00036 };
00037
00038 class Studentas : public Zmogus {
00039     private:
00040         Vector<int> nd_;
00041         int egz_;
00042         double balas_;
00043         double vid_;
00044         double mediana_;
00045     public:
00046         Studentas() : egz_(0), balas_(0), vid_(0), mediana_(0) {} // default konstruktorius
00047
00048         Studentas(const string& vardas, const string& pavarde, const Vector<int>& nd, int egz, double
00049         balas, double vid, double mediana) // konstruktorius
00050             : Zmogus(vardas, pavarde), nd_(nd), egz_(egz), balas_(balas), vid_(vid), mediana_(mediana) {}
00051         ~Studentas() {nd_.clear();} // destruktorius
00052
00053         Studentas(const Studentas& other)
00054             : Zmogus(other.getVardas(), other.getPavarde()), nd_(other.nd_), egz_(other.egz_),
00055             balas_(other.balas_), vid_(other.vid_), mediana_(other.mediana_) {} // copy constructor
00056
00057         Studentas& operator=(const Studentas& other) { // copy assignment operatorius
00058             if (this != &other) {
00059                 Zmogus::setVardas(other.getVardas());
```

```

00059         Zmogus::setPavarde(other.getPavarde());
00060         nd_ = other.nd_;
00061         egz_ = other.egz_;
00062         balas_ = other.balas_;
00063         vid_ = other.vid_;
00064         mediana_ = other.mediana_;
00065     }
00066     return *this;
00067 }
00068
00069 Studentas(Studentas&& other) noexcept
00070 { Zmogus::setVardas(other.getVardas());
00071   Zmogus::setPavarde(other.getPavarde());
00072   nd_ = other.nd_;
00073   egz_ = other.egz_;
00074   balas_ = other.balas_;
00075   vid_ = other.vid_;
00076   mediana_ = mediana_;
00077
00078   other.setVardas("");
00079   other.setPavarde("");
00080   other.nd_.clear();
00081   other.egz_ = 0;
00082   other.balas_ = 0;
00083   other.vid_ = 0;
00084   other.mediana_ = 0;
00085
00086   //cout << "Move konstruktorius suveike" << endl;
00087 };
00088
00089 Studentas& operator=(Studentas&& other) noexcept { // move assignment operatorius
00090     if (this != &other) {
00091         Zmogus::setVardas(move(other.getVardas()));
00092         Zmogus::setPavarde(move(other.getPavarde()));
00093         nd_ = move(other.nd_);
00094         egz_ = other.egz_;
00095         balas_ = other.balas_;
00096         vid_ = other.vid_;
00097         mediana_ = other.mediana_;
00098         other.egz_ = 0;
00099         other.balas_ = 0;
00100         other.vid_ = 0;
00101         other.mediana_ = 0;
00102     }
00103     return *this;
00104 }
00105
00106 // Getter and setter functions
00107 inline string getVardas() const override {return vardas_;}
00108 inline string getPavarde() const override {return pavarde_;}
00109 inline Vector<int>& getNd() { return nd_; }
00110 inline int getEgz() const { return egz_; }
00111 inline double getBalas() const { return balas_; }
00112 inline double getVid() const { return vid_; }
00113 inline double getMediana() const { return mediana_; }
00114 inline void setVardas(const string& vardas) { vardas_ = vardas; }
00115 inline void setPavarde(const string& pavarde) { pavarde_ = pavarde; }
00116 inline void setNd(const Vector<int>& nd) { nd_ = nd; }
00117 inline void setEgz(const int& egz) { egz_ = egz; }
00118 inline void setBalas(const double& balas) { balas_ = balas; }
00119 inline void setVid(const double& vid) { vid_ = vid; }
00120 inline void setMediana(const double& mediana) { mediana_ = mediana; }
00121
00122 friend istream& operator>(istream& is, Studentas& student) { // input metodus
00123     cout << "Enter student's name and surname: ";
00124     is >> student.vardas_ >> student.pavarde_;
00125
00126     cout << "Enter student's exam score: ";
00127     is >> student.egz_;
00128
00129     cout << "Enter student's homework scores (enter -1 to stop): ";
00130     int score;
00131     while (is >> score && score != -1) {
00132         student.nd_.push_back(score);
00133     }
00134
00135     return is;
00136 }
00137
00138 friend ostream& operator<(ostream& os, const Studentas& student) { // output metodus
00139     os << "Name: " << student.vardas_ << " " << student.pavarde_ << endl;
00140     os << "Exam score: " << student.egz_ << endl;
00141     os << "Homework scores: ";
00142     for (int score : student.nd_) {
00143         os << score << " ";
00144     }
00145 }

```

```

00146         os << endl;
00147         os << "Final score: " << student.balas_ << endl;
00148         os << "Average: " << student.vid_ << endl;
00149         os << "Median: " << student.mediana_ << endl;
00150         return os;
00151     }
00152 };
00153
00154 void skaitymas(Vector<Studentas>&, int);
00155 void skaitymasTeksto(Vector<Studentas>&);
00156 bool pagalVarda(const Studentas&, const Studentas&);
00157 bool pagalPavarde(const Studentas&, const Studentas&);
00158 bool pagalVidurki(const Studentas&, const Studentas&);
00159 bool pagalMediana(const Studentas&, const Studentas&);
00160 void spausdinti(const Vector<Studentas>&);
00161 void spausdintiTeksto(const Vector<Studentas>&);
00162 void skaiciavimas(Vector<Studentas>&);
00163 void padalintiStudentus(Vector<Studentas>&);
00164 void generavimasPazymiu(Vector<Studentas>&, int n);
00165 void generavimasPazymiuCase2(Vector<Studentas>&);
00166 void generavimasStudentu(Vector<Studentas>&, int);
00167 void generavimasFailo(Vector<Studentas>&, int);
00168 bool isValidName(const string&);
00169 void test_constructor ();
00170 void test_copy_constructor ();
00171 void test_move_constructor ();
00172 void test_copy_assignment ();
00173 void test_move_assignment ();
00174 void test_input_operator ();
00175 void test_output_operator ();
00176 #endif

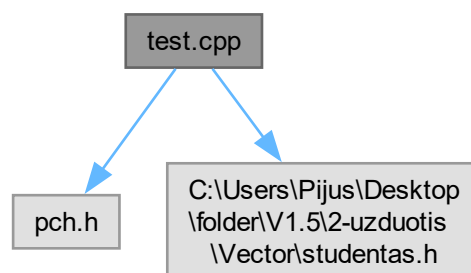
```

## 5.7 test.cpp File Reference

```
#include "pch.h"
```

```
#include "C:\Users\Pijus\Desktop\folder\V1.5\2-uzduotis\Vector\studentas.h"
```

Include dependency graph for test.cpp:



### Functions

- TEST (StudentTest, ConstructorTest)
- TEST (StudentTest, CopyConstructor)
- TEST (StudentTest, MoveConstructor)
- TEST (StudentTest, CopyAssignment)
- TEST (StudentTest, MoveAssignment)

## 5.7.1 Function Documentation

### 5.7.1.1 TEST() [1/5]

```
TEST (
    StudentTest ,
    ConstructorTest )
```

### 5.7.1.2 TEST() [2/5]

```
TEST (
    StudentTest ,
    CopyAssignment )
```

### 5.7.1.3 TEST() [3/5]

```
TEST (
    StudentTest ,
    CopyConstructor )
```

### 5.7.1.4 TEST() [4/5]

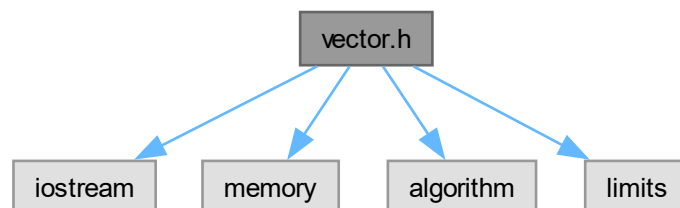
```
TEST (
    StudentTest ,
    MoveAssignment )
```

### 5.7.1.5 TEST() [5/5]

```
TEST (
    StudentTest ,
    MoveConstructor )
```

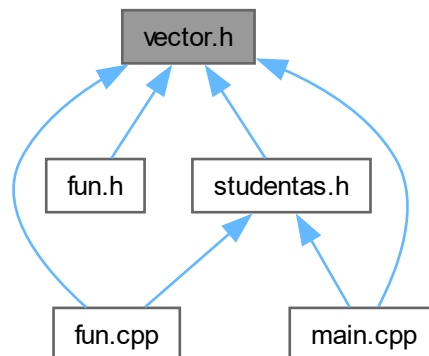
## 5.8 vector.h File Reference

```
#include <iostream>
#include <memory>
#include <algorithm>
#include <limits>
Include dependency graph for vector.h:
```





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



## Classes

- class `Vector< T >`

## 5.9 vector.h

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

```

00001 #ifndef TA
00002 #define TA
00003
00004 #include <iostream>
00005 #include <memory>
00006 #include <algorithm>
00007 #include <limits>
00008
00009 template <typename T>
00010
00011 class Vector{
00012 public:
00013     typedef T value_type;
00014     typedef size_t size_type;
00015     typedef T& reference;
00016     typedef const T& const_reference;
00017     typedef T* iterator;
00018     typedef const T* const_iterator;
00019
00020
00021     Vector() {create ();}
00022     explicit Vector (size_type n, const T& t = T{}) {create (n, t);}
00023     Vector (const Vector& v) {create(v.begin(), v.end());}
00024     template <class InputIterator>
00025     Vector (InputIterator first, InputIterator last) {create(first, last);}
00026     Vector (Vector && v) {create(); swap(v); v.uncreate();}
00027     Vector (const std::initializer_list<T> il) {create(il.begin(), il.end());}
00028     ~Vector() {uncreate();}
00029     operator std::vector<T> () const{
00030         return std::vector<T>(dat, avail);
00031     }
00032     Vector& operator = (const Vector& other) {
00033         if(this != &other){
00034             uncreate();
00035             create(other.begin(), other.end());
00036         }
00037         return *this;
00038     };
00039     Vector& operator = (Vector&& other){
00040         if (this != &other){

```

```

00052         create(other.begin(), other.end());
00053         uncreate();
00054     }
00055     return *this;
00056 };
00057 template <class InputIterator>
00058 void assign (InputIterator first, InputIterator last){
00059     uncreate();
00060     create(first, last);
00061 }
00062 void assign (size_type n, const value_type& val){
00063     uncreate();
00064     create(n, val);
00065 }
00066 void assign (std::initializer_list <value_type> il){
00067     uncreate ();
00068     create (il);
00069 }
00070 const_reference at (size_type n) const {
00071     if (n >= size() || n < 0){
00072         throw std::out_of_range("Index Out of Range");
00073     }
00074     return dat[n];
00075 }
00076 }
00077 T& operator[] (size_type n) {return dat[n];}
00078 const T& operator [] (size_type n) const { return dat[n];}
00079 reference at (size_type n){
00080     if (n >= size() || n < 0){
00081         throw std::out_of_range("Index Out of Range");
00082     }
00083     return dat[n];
00084 }
00085 }
00086 reference front(){
00087     return dat[0];
00088 };
00089 const_reference front() const{
00090     return dat[0];
00091 }
00092 reference back(){
00093     return dat[size() - 1];
00094 }
00095 const_reference back() const{
00096     return dat[size() - 1];
00097 }
00098 value_type* data() noexcept{
00099     return dat;
00100 }
00101 const value_type* data() const noexcept{
00102     return dat;
00103 }
00104 iterator begin() {return dat;}
00105 const_iterator begin() const {return dat;}
00106 iterator end() {return avail;}
00107 const_iterator end() const {return avail;}
00108 size_type size() const {return avail-dat;}
00109 size_type max_size() const {return std::numeric_limits<size_type>::max();}
00110 void resize(size_type sz){
00111     if (sz < size()) {
00112         iterator it = dat + sz;
00113         while (it != avail) {
00114             alloc.destroy(it++);
00115         }
00116         avail = dat + sz;
00117     }
00118     else if (sz > capacity()) {
00119         grow(sz);
00120         std::uninitialized_fill(avail, dat + sz, value_type());
00121         avail = dat + sz;
00122     }
00123     else if (sz > size()) {
00124         std::uninitialized_fill(avail, dat + sz, value_type());
00125         avail = dat + sz;
00126     }
00127 }
00128 void resize(size_type sz, const value_type& value) {
00129     if (sz > capacity()) {
00130         grow(sz);
00131     }
00132     if (sz > size()) {
00133         insert(end(), sz - size(), value);
00134     } else if (sz < size()) {
00135         avail = dat + sz;
00136     }
00137 }
00138 size_type capacity() const {return limit-dat;}
00139 bool empty() const noexcept { return size() == 0;}

```

```

00143     void reserve (size_type n) {
00144         if (n > capacity()) {
00145             grow(n);
00146         }
00147     }
00148     void shrink_to_fit(){
00149         if (limit > avail)
00150             limit = avail;
00151     }
00153     void clear () noexcept {uncreate();}
00154     iterator insert (const_iterator position, const value_type& val){return insert(position, 1, val);}
00155     iterator insert(iterator position, size_type n, const value_type& val) {
00156         if (position < begin() || position > end()) {
00157             throw std::out_of_range("Index out of range");
00158         }
00159         if (avail + n > limit) {
00160             size_type index = position - begin();
00161             grow(n);
00162             position = begin() + index;
00163         }
00164         for (iterator it = end() + n - 1; it != position + n - 1; --it) {
00165             *it = std::move(*(it - n));
00166         }
00167         std::uninitialized_fill(position, position + n, val);
00168         avail += n;
00169         return position;
00170     }
00171     iterator erase(iterator position) {
00172         if (position < dat || position > avail) {
00173             throw std::out_of_range("Index out of range");
00174         }
00175         std::move(position + 1, avail, position);
00176         alloc.destroy(avail - 1);
00177         --avail;
00178         return position;
00179     }
00180     iterator erase(iterator first, iterator last) {
00181         iterator new_available = std::uninitialized_copy(last, avail, first);
00182         iterator it = avail;
00183         while (it != new_available) {
00184             alloc.destroy(--it);
00185         }
00186         avail = new_available;
00187         return last;
00188     }
00189     void push_back (const value_type& t) {
00190         if (avail == limit)
00191             grow();
00192         unchecked_append(t);
00193     }
00194     void push_back (value_type&& val) {
00195         if (avail == limit)
00196             grow();
00197         unchecked_append(val);
00198     }
00199     void pop_back() {
00200         if (avail != dat)
00201             alloc.destroy(--avail);
00202     }
00203     void swap(Vector& x) {
00204         std::swap(dat, x.dat);
00205         std::swap(avail, x.avail);
00206         std::swap(limit, x.limit);
00207     }
00208     bool operator== (const Vector<T>& other) const{
00209         if (size() != other.size()){
00210             return false;
00211         }
00212         return std::equal(begin(), end(), other.begin());
00213     }
00214     bool operator!= (const Vector<T>& other) const{
00215         return !(*this == other);
00216     }
00217     bool operator< (const Vector<T>& other) const{
00218         return std::lexicographical_compare(begin(), end(), other.begin(), other.end());
00219     }
00220     bool operator<= (const Vector<T>& other) const{
00221         return !(other < *this);
00222     }

```

```

00232     }
00233     bool operator> (const Vector<T>& other) const{
00234         return std::lexicographical_compare(other.begin(), other.end(), begin(), end());
00235     }
00236     bool operator>= (const Vector<T>& other) const{
00237         return !(other > *this);
00238     }
00239     void swap (Vector<T>& x, Vector<T>& y) {
00240         std::swap(x,y);
00241     }
00242 private:
00243     iterator dat;
00244     iterator avail;
00245     iterator limit;
00246     std::allocator<T> alloc;
00247
00248     void create() {dat = avail = limit = nullptr;}
00249     void create (size_type n, const T& val) {
00250         dat = alloc.allocate(n);
00251         limit = avail = dat + n;
00252         std::uninitialized_fill(dat, limit, val);
00253     }
00254     void create(const_iterator i, const_iterator j) {
00255         dat = alloc.allocate(j - i);
00256         limit = avail = std::uninitialized_copy(i, j, dat);
00257     }
00258     void uncreate(){
00259         if (dat) {
00260             iterator it = avail;
00261             while (it != dat) {
00262                 alloc.destroy(--it);
00263             }
00264             alloc.deallocate(dat, limit - dat);
00265         }
00266         dat = limit = avail = nullptr;
00267     }
00268     void grow(size_type new_capacity = 1) {
00269         size_type new_size = std::max(new_capacity, 2 * capacity());
00270         iterator new_data = alloc.allocate(new_size);
00271         iterator new_avail = std::uninitialized_copy(dat, avail, new_data);
00272         uncreate();
00273         dat = new_data;
00274         avail = new_avail;
00275         limit = dat + new_size;
00276     }
00277
00278     void unchecked_append(const T& val) {
00279         alloc.construct(avail++, val);
00280     }
00281 };
00282
00283
00284 #endif

```

# Index

- ~Studentas
  - Studentas, [10](#)
- ~Vector
  - Vector< T >, [16](#)
- ~Zmogus
  - Zmogus, [23](#)
- assign
  - Vector< T >, [16](#), [17](#)
- at
  - Vector< T >, [17](#)
- back
  - Vector< T >, [17](#)
- balas
  - studentas, [7](#)
- begin
  - Vector< T >, [17](#)
- capacity
  - Vector< T >, [18](#)
- clear
  - Vector< T >, [18](#)
- const\_iterator
  - Vector< T >, [14](#)
- const\_reference
  - Vector< T >, [14](#)
- data
  - Vector< T >, [18](#)
- egz
  - studentas, [7](#)
- empty
  - Vector< T >, [18](#)
- end
  - Vector< T >, [18](#)
- erase
  - Vector< T >, [18](#), [19](#)
- front
  - Vector< T >, [19](#)
- fun.cpp, [25](#)
  - generavimasFailo, [26](#)
  - generavimasPazymiu, [26](#)
  - generavimasPazymiuCase2, [26](#)
  - generavimasStudentu, [26](#)
  - isValidName, [26](#)
  - padalintiStudentus, [26](#)
  - pagalMediana, [26](#)
  - pagalPavarde, [26](#)
- generavimasFailo
  - pagalVarda, [26](#)
  - pagalVidurki, [27](#)
  - skaiciavimas, [27](#)
  - skaitymas, [27](#)
  - skaitymasTeksto, [27](#)
  - spausdinti, [27](#)
  - spausdintiTeksto, [27](#)
  - test\_constructor, [27](#)
  - test\_copy\_assignment, [27](#)
  - test\_copy\_constructor, [27](#)
  - test\_input\_operator, [28](#)
  - test\_move\_assignment, [28](#)
  - test\_move\_constructor, [28](#)
  - test\_output\_operator, [28](#)
- fun.h, [28](#)
  - generavimasFailo, [29](#)
  - generavimasPazymiu, [29](#)
  - generavimasPazymiuCase2, [29](#)
  - generavimasStudentu, [29](#)
  - isValidName, [30](#)
  - MAX\_ND\_SIZE, [31](#)
  - MAX\_STUDENTS, [31](#)
  - padalintiStudentus, [30](#)
  - pagalMediana, [30](#)
  - pagalPavarde, [30](#)
  - pagalVarda, [30](#)
  - pagalVidurki, [30](#)
  - skaiciavimas, [30](#)
  - skaitymas, [30](#)
  - skaitymasTeksto, [31](#)
  - spausdinti, [31](#)
  - spausdintiTeksto, [31](#)
- generavimasFailo
  - fun.cpp, [26](#)
  - fun.h, [29](#)
  - studentas.h, [34](#)
- generavimasPazymiu
  - fun.cpp, [26](#)
  - fun.h, [29](#)
  - studentas.h, [34](#)
- generavimasPazymiuCase2
  - fun.cpp, [26](#)
  - fun.h, [29](#)
  - studentas.h, [34](#)
- generavimasStudentu
  - fun.cpp, [26](#)
  - fun.h, [29](#)
  - studentas.h, [34](#)
- getBalas

- Studentas, 10
- getEgz
  - Studentas, 10
- getMediana
  - Studentas, 11
- getNd
  - Studentas, 11
- getPavarde
  - Studentas, 11
  - Zmogus, 24
- getVardas
  - Studentas, 11
  - Zmogus, 24
- getVid
  - Studentas, 11
- insert
  - Vector< T >, 19
- isValidName
  - fun.cpp, 26
  - fun.h, 30
  - studentas.h, 34
- iterator
  - Vector< T >, 15
- main
  - main.cpp, 32
- main.cpp, 32
  - main, 32
- MAX\_ND\_SIZE
  - fun.h, 31
  - studentas.h, 37
- max\_size
  - Vector< T >, 19
- MAX\_STUDENTS
  - fun.h, 31
  - studentas.h, 37
- mediana
  - studentas, 8
- nd
  - studentas, 8
- operator std::vector< T >
  - Vector< T >, 19
- operator!=
  - Vector< T >, 20
- operator<
  - Vector< T >, 20
- operator<<
  - Studentas, 12
- operator<=
  - Vector< T >, 20
- operator>
  - Vector< T >, 20
- operator>>
  - Studentas, 12
- operator>=
  - Vector< T >, 21
- operator=
  - Studentas, 11
  - Vector< T >, 20
- operator==
  - Vector< T >, 20
- operator[]
  - Vector< T >, 21
- padalintiStudentus
  - fun.cpp, 26
  - fun.h, 30
  - studentas.h, 35
- pagalMediana
  - fun.cpp, 26
  - fun.h, 30
  - studentas.h, 35
- pagalPavarde
  - fun.cpp, 26
  - fun.h, 30
  - studentas.h, 35
- pagalVarda
  - fun.cpp, 26
  - fun.h, 30
  - studentas.h, 35
- pagalVidurki
  - fun.cpp, 27
  - fun.h, 30
  - studentas.h, 35
- pavarde
  - studentas, 8
- pavarde\_
  - Zmogus, 24
- pop\_back
  - Vector< T >, 21
- push\_back
  - Vector< T >, 21
- reference
  - Vector< T >, 15
- reserve
  - Vector< T >, 21
- resize
  - Vector< T >, 21, 22
- setBalas
  - Studentas, 11
- setEgz
  - Studentas, 11
- setMediana
  - Studentas, 12
- setNd
  - Studentas, 12
- setPavarde
  - Studentas, 12
  - Zmogus, 24
- setVardas
  - Studentas, 12
  - Zmogus, 24
- setVid

- Studentas, 12
- shrink\_to\_fit
  - Vector< T >, 22
- size
  - Vector< T >, 22
- size\_type
  - Vector< T >, 15
- skaiciavimas
  - fun.cpp, 27
  - fun.h, 30
  - studentas.h, 35
- skaitymas
  - fun.cpp, 27
  - fun.h, 30
  - studentas.h, 35
- skaitymasTeksto
  - fun.cpp, 27
  - fun.h, 31
  - studentas.h, 35
- spausdinti
  - fun.cpp, 27
  - fun.h, 31
  - studentas.h, 36
- spausdintiTeksto
  - fun.cpp, 27
  - fun.h, 31
  - studentas.h, 36
- Studentas, 8
  - ~Studentas, 10
  - getBalas, 10
  - getEgz, 10
  - getMediana, 11
  - getNd, 11
  - getPavarde, 11
  - getVardas, 11
  - getVid, 11
  - operator<<, 12
  - operator>>, 12
  - operator=, 11
  - setBalas, 11
  - setEgz, 11
  - setMediana, 12
  - setNd, 12
  - setPavarde, 12
  - setVardas, 12
  - setVid, 12
  - Studentas, 10
- studentas, 7
  - balas, 7
  - egz, 7
  - mediana, 8
  - nd, 8
  - pavarde, 8
  - vardas, 8
  - vid, 8
- studentas.h, 33
  - generavimasFailo, 34
  - generavimasPazymiu, 34
  - generavimasPazymiuCase2, 34
  - generavimasStudentu, 34
  - isValidName, 34
  - MAX\_ND\_SIZE, 37
  - MAX\_STUDENTS, 37
  - padalintiStudentus, 35
  - pagalMediana, 35
  - pagalPavarde, 35
  - pagalVarda, 35
  - pagalVidurki, 35
  - skaiciavimas, 35
  - skaitymas, 35
  - skaitymasTeksto, 35
  - spausdinti, 36
  - spausdintiTeksto, 36
  - test\_constructor, 36
  - test\_copy\_assignment, 36
  - test\_copy\_constructor, 36
  - test\_input\_operator, 36
  - test\_move\_assignment, 36
  - test\_move\_constructor, 36
  - test\_output\_operator, 36
- swap
  - Vector< T >, 22
- TEST
  - test.cpp, 40
- test.cpp, 39
  - TEST, 40
- test\_constructor
  - fun.cpp, 27
  - studentas.h, 36
- test\_copy\_assignment
  - fun.cpp, 27
  - studentas.h, 36
- test\_copy\_constructor
  - fun.cpp, 27
  - studentas.h, 36
- test\_input\_operator
  - fun.cpp, 28
  - studentas.h, 36
- test\_move\_assignment
  - fun.cpp, 28
  - studentas.h, 36
- test\_move\_constructor
  - fun.cpp, 28
  - studentas.h, 36
- test\_output\_operator
  - fun.cpp, 28
  - studentas.h, 36
- value\_type
  - Vector< T >, 15
- vardas
  - studentas, 8
- vardas\_
  - Zmogus, 24
- Vector
  - Vector< T >, 15, 16

Vector< T >, 13  
    ~Vector, 16  
    assign, 16, 17  
    at, 17  
    back, 17  
    begin, 17  
    capacity, 18  
    clear, 18  
    const\_iterator, 14  
    const\_reference, 14  
    data, 18  
    empty, 18  
    end, 18  
    erase, 18, 19  
    front, 19  
    insert, 19  
    iterator, 15  
    max\_size, 19  
    operator std::vector< T >, 19  
    operator!=, 20  
    operator<, 20  
    operator<=, 20  
    operator>, 20  
    operator>=, 21  
    operator=, 20  
    operator==, 20  
    operator[], 21  
    pop\_back, 21  
    push\_back, 21  
    reference, 15  
    reserve, 21  
    resize, 21, 22  
    shrink\_to\_fit, 22  
    size, 22  
    size\_type, 15  
    swap, 22  
    value\_type, 15  
    Vector, 15, 16  
vector.h, 40  
vid  
    studentas, 8

Zmogus, 23  
    ~Zmogus, 23  
    getPavarde, 24  
    getVardas, 24  
    pavarde\_, 24  
    setPavarde, 24  
    setVardas, 24  
    vardas\_, 24  
    Zmogus, 23