

## My Project

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 Class Reference	7
4.1.1 Constructor & Destructor Documentation	8
4.1.1.1 Studentas() [1/4]	8
4.1.1.2 Studentas() [2/4]	8
4.1.1.3 Studentas() [3/4]	8
4.1.1.4 Studentas() [4/4]	9
4.1.1.5 ~Studentas()	9
4.1.2 Member Function Documentation	9
4.1.2.1 addPazymys()	9
4.1.2.2 getEgzaminoRezultatas()	9
4.1.2.3 getGalutinisVid()	9
4.1.2.4 getMediana()	9
4.1.2.5 getPavarde()	9
4.1.2.6 getPazymiai()	9
4.1.2.7 getVardas()	9
4.1.2.8 operator=() [1/2]	10
4.1.2.9 operator=() [2/2]	10
4.1.2.10 setEgzaminoRezultatas()	10
4.1.2.11 setGalutinisVid()	10
4.1.2.12 setMediana()	10
4.1.2.13 setPavarde()	10
4.1.2.14 setPazymiai()	10
4.1.2.15 setVardas()	10
4.1.3 Friends And Related Symbol Documentation	11
4.1.3.1 operator<<	11
4.1.3.2 operator>>	11
4.1.4 Member Data Documentation	11
4.1.4.1 egzamino_rezultatas_	11
4.1.4.2 galutinis_vid_	11
4.1.4.3 mediana_	11
4.1.4.4 pazymiai_	11
4.2 Vector< T > Class Template Reference	11
4.2.1 Member Typedef Documentation	13

4.2.1.1 <code>const_iterator</code>	13
4.2.1.2 <code>const_reference</code>	13
4.2.1.3 <code>iterator</code>	14
4.2.1.4 <code>reference</code>	14
4.2.1.5 <code>size_type</code>	14
4.2.1.6 <code>value_type</code>	14
4.2.2 Constructor & Destructor Documentation	14
4.2.2.1 <code>Vector()</code> [1/6]	14
4.2.2.2 <code>Vector()</code> [2/6]	14
4.2.2.3 <code>Vector()</code> [3/6]	15
4.2.2.4 <code>Vector()</code> [4/6]	15
4.2.2.5 <code>Vector()</code> [5/6]	15
4.2.2.6 <code>Vector()</code> [6/6]	15
4.2.2.7 <code>~Vector()</code>	15
4.2.3 Member Function Documentation	15
4.2.3.1 <code>assign()</code> [1/3]	15
4.2.3.2 <code>assign()</code> [2/3]	16
4.2.3.3 <code>assign()</code> [3/3]	16
4.2.3.4 <code>at()</code> [1/2]	16
4.2.3.5 <code>at()</code> [2/2]	16
4.2.3.6 <code>back()</code> [1/2]	16
4.2.3.7 <code>back()</code> [2/2]	16
4.2.3.8 <code>begin()</code> [1/2]	16
4.2.3.9 <code>begin()</code> [2/2]	16
4.2.3.10 <code>capacity()</code>	17
4.2.3.11 <code>clear()</code>	17
4.2.3.12 <code>create()</code> [1/3]	17
4.2.3.13 <code>create()</code> [2/3]	17
4.2.3.14 <code>create()</code> [3/3]	17
4.2.3.15 <code>data()</code> [1/2]	17
4.2.3.16 <code>data()</code> [2/2]	17
4.2.3.17 <code>empty()</code>	17
4.2.3.18 <code>end()</code> [1/2]	18
4.2.3.19 <code>end()</code> [2/2]	18
4.2.3.20 <code>erase()</code> [1/2]	18
4.2.3.21 <code>erase()</code> [2/2]	18
4.2.3.22 <code>front()</code> [1/2]	18
4.2.3.23 <code>front()</code> [2/2]	18
4.2.3.24 <code>grow()</code>	18
4.2.3.25 <code>insert()</code> [1/2]	18
4.2.3.26 <code>insert()</code> [2/2]	19
4.2.3.27 <code>max_size()</code>	19

4.2.3.28 operator!=( )	19
4.2.3.29 operator<( )	19
4.2.3.30 operator<=( )	19
4.2.3.31 operator==( 1/2 )	19
4.2.3.32 operator==( 2/2 )	19
4.2.3.33 operator==( )	20
4.2.3.34 operator>( )	20
4.2.3.35 operator>=( )	20
4.2.3.36 operator[] ( 1/2 )	20
4.2.3.37 operator[] ( 2/2 )	20
4.2.3.38 pop_back( )	20
4.2.3.39 push_back( 1/2 )	20
4.2.3.40 push_back( 2/2 )	20
4.2.3.41 reserve( )	21
4.2.3.42 resize( 1/2 )	21
4.2.3.43 resize( 2/2 )	21
4.2.3.44 shrink_to_fit( )	21
4.2.3.45 size( )	21
4.2.3.46 swap( 1/2 )	21
4.2.3.47 swap( 2/2 )	21
4.2.3.48 unchecked_append( )	21
4.2.3.49 uncreate( )	22
4.2.4 Member Data Documentation	22
4.2.4.1 alloc	22
4.2.4.2 avail	22
4.2.4.3 dat	22
4.2.4.4 limit	22
4.3 Zmogus Class Reference	22
4.3.1 Constructor & Destructor Documentation	23
4.3.1.1 Zmogus( 1/2 )	23
4.3.1.2 Zmogus( 2/2 )	23
4.3.1.3 ~Zmogus( )	23
4.3.2 Member Function Documentation	23
4.3.2.1 getPavarde( )	23
4.3.2.2 getVardas( )	23
4.3.2.3 setPavarde( )	24
4.3.2.4 setVardas( )	24
4.3.3 Member Data Documentation	24
4.3.3.1 pavarde_	24
4.3.3.2 vardas_	24
<b>5 File Documentation</b>	<b>25</b>

5.1 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/Funkcijos.cpp File Reference	25
5.1.1 Function Documentation	26
5.1.1.1 ar_vargsiukas()	26
5.1.1.2 failolsvedimasDeque1()	26
5.1.1.3 failolsvedimasList1()	26
5.1.1.4 failolsvedimasVector1()	26
5.1.1.5 generuotiFaila()	26
5.1.1.6 mediana()	26
5.1.1.7 nuskaitytiFailaDeque1()	27
5.1.1.8 nuskaitytiFailaDeque2()	27
5.1.1.9 nuskaitytiFailaDeque3()	27
5.1.1.10 nuskaitytiFailaList1()	27
5.1.1.11 nuskaitytiFailaList2()	27
5.1.1.12 nuskaitytiFailaList3()	27
5.1.1.13 nuskaitytiFailaVector1()	27
5.1.1.14 nuskaitytiFailaVector2()	28
5.1.1.15 nuskaitytiFailaVector3()	28
5.1.1.16 palyginti_pagal_galutini_vidurki()	28
5.1.1.17 palyginti_pagal_galutini_vidurki_didejimo_tvarka()	28
5.1.1.18 palyginti_pagal_mediana()	28
5.1.1.19 palyginti_pagal_pavarde()	28
5.1.1.20 palyginti_pagal_varda()	28
5.1.1.21 RezultatuVaizdavimas()	28
5.1.1.22 testai()	29
5.1.1.23 vidurkis_galutinis()	29
5.2 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/funkcijos.h File Reference	29
5.2.1 Function Documentation	30
5.2.1.1 ar_vargsiukas()	30
5.2.1.2 failolsvedimasDeque1()	30
5.2.1.3 failolsvedimasList1()	30
5.2.1.4 failolsvedimasVector1()	30
5.2.1.5 generuotiFaila()	30
5.2.1.6 mediana()	30
5.2.1.7 nuskaitytiFailaDeque1()	30
5.2.1.8 nuskaitytiFailaDeque2()	31
5.2.1.9 nuskaitytiFailaDeque3()	31
5.2.1.10 nuskaitytiFailaList1()	31
5.2.1.11 nuskaitytiFailaList2()	31
5.2.1.12 nuskaitytiFailaList3()	31
5.2.1.13 nuskaitytiFailaVector1()	31
5.2.1.14 nuskaitytiFailaVector2()	31
5.2.1.15 nuskaitytiFailaVector3()	32

5.2.1.16 palyginti_pagal_galutini_vidurki()	32
5.2.1.17 palyginti_pagal_galutini_vidurki_didejimo_tvarka()	32
5.2.1.18 palyginti_pagal_mediana()	32
5.2.1.19 palyginti_pagal_pavarde()	32
5.2.1.20 palyginti_pagal_varda()	32
5.2.1.21 RezultatuVaizdavimas()	32
5.2.1.22 testai()	32
5.2.1.23 vidurkis_galutinis()	33
5.3 funkcijos.h	33
5.4 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/studentas.h File Reference	33
5.5 studentas.h	34
5.6 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/vector.h File Reference	35
5.7 vector.h	35
5.8 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/Vektoriai.cpp File Reference	39
5.8.1 Function Documentation	40
5.8.1.1 main()	40
5.8.2 Variable Documentation	40
5.8.2.1 m	40
5.8.2.2 MAX_ND	40
5.8.2.3 MAX_STUDENTU	40
5.8.2.4 menu	40
5.8.2.5 n	40
5.8.2.6 pasirinkimas	40
5.8.2.7 raide	41
5.8.2.8 suma	41
5.8.2.9 variantas_namu_darbas	41
5.8.2.10 variantas_studentas	41
5.8.2.11 vidurkis	41
<b>Index</b>	<b>43</b>





# 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 . . . . .	22
Studentas . . . . .	7



## 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="#">Vector&lt; T &gt;</a>	11
<a href="#">Zmogus</a>	22



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

C:/Users/acer/Desktop/Adri/VSCode/Programavimas/ <a href="#">Funkcijos.cpp</a>	25
C:/Users/acer/Desktop/Adri/VSCode/Programavimas/ <a href="#">funkcijos.h</a>	29
C:/Users/acer/Desktop/Adri/VSCode/Programavimas/ <a href="#">studentas.h</a>	33
C:/Users/acer/Desktop/Adri/VSCode/Programavimas/ <a href="#">vector.h</a>	35
C:/Users/acer/Desktop/Adri/VSCode/Programavimas/ <a href="#">Vektoriai.cpp</a>	39



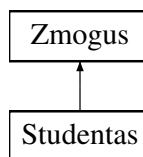
## Chapter 4

# Class Documentation

### 4.1 Studentas Class Reference

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

Inheritance diagram for Studentas:



#### Public Member Functions

- `Studentas ()`
- `Studentas (const Vector< int > &pazymiai, const string &vardas, const string &pavarde, int egzamino_rezultatas, double galutinis_vid, double mediana)`
- `Studentas (const Studentas &other)`
- `Studentas & operator= (const Studentas &other)`
- `Studentas (Studentas &&other) noexcept`
- `Studentas & operator= (Studentas &&other) noexcept`
- `~Studentas ()`
- `void setVardas (const string &vardas) override`
- `void setPavarde (const string &pavarde) override`
- `void setPazymiai (const Vector< int > &pazymiai)`
- `void setEgzaminoRezultatas (int egzamino_rezultatas)`
- `void setGalutinisVid (double galutinis_vid)`
- `void setMediana (double mediana)`
- `string getVardas () const override`
- `string getPavarde () const override`
- `const Vector< int > & getPazymiai () const`
- `int getEgzaminoRezultatas () const`
- `double getGalutinisVid () const`
- `double getMediana () const`
- `void addPazymys (int pazymys)`

## Public Member Functions inherited from [Zmogus](#)

- [Zmogus](#) ()=default
- [Zmogus](#) (const string &vardas, const string &pavarde)
- virtual [~Zmogus](#) ()

## Private Attributes

- [Vector](#)< int > [pazymiai\\_](#)
- int [egzamino\\_rezultatas\\_](#)
- double [galutinis\\_vid\\_](#)
- double [mediana\\_](#)

## Friends

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

## Additional Inherited Members

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

- string [vardas\\_](#)
- 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 Vector< int > & pazymiai,
    const string & vardas,
    const string & pavarde,
    int egzamino_rezultatas,
    double galutinis_vid,
    double mediana ) [inline]
```

### 4.1.1.3 [Studentas\(\)](#) [3/4]

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



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

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

#### 4.1.1.5 ~Studentas()

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

### 4.1.2 Member Function Documentation

#### 4.1.2.1 addPazymys()

```
void Studentas::addPazymys (
    int pazymys ) [inline]
```

#### 4.1.2.2 getEgzaminoRezultatas()

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

#### 4.1.2.3 getGalutinisVid()

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

#### 4.1.2.4 getMediana()

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

#### 4.1.2.5 getPavarde()

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

Implements [Zmogus](#).

#### 4.1.2.6 getPazymiai()

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

#### 4.1.2.7 getVardas()

```
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 setEgzaminoRezultatas()

```
void Studentas::setEgzaminoRezultatas (
    int egzamino_rezultatas ) [inline]
```

#### 4.1.2.11 setGalutinisVid()

```
void Studentas::setGalutinisVid (
    double galutinis_vid ) [inline]
```

#### 4.1.2.12 setMediana()

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

#### 4.1.2.13 setPavarde()

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

Reimplemented from [Zmogus](#).

#### 4.1.2.14 setPazymiai()

```
void Studentas::setPazymiai (
    const Vector< int > & pazymiai ) [inline]
```

#### 4.1.2.15 setVardas()

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

Reimplemented from [Zmogus](#).

### 4.1.3 Friends And Related Symbol Documentation

#### 4.1.3.1 operator<<

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

#### 4.1.3.2 operator>>

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

### 4.1.4 Member Data Documentation

#### 4.1.4.1 egzamino\_rezultatas\_

```
int Studentas::egzamino_rezultatas_ [private]
```

#### 4.1.4.2 galutinis\_vid\_

```
double Studentas::galutinis_vid_ [private]
```

#### 4.1.4.3 mediana\_

```
double Studentas::mediana_ [private]
```

#### 4.1.4.4 pazymiai\_

```
Vector<int> Studentas::pazymiai_ [private]
```

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

- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/studentas.h

## 4.2 Vector< T > Class Template Reference

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

## Public Types

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

## Public Member Functions

- [Vector](#) ()  
*Konstruktoriai /default.*
- [Vector](#) ([size\\_type](#) n, const T &t=T{})
- [Vector](#) (const [Vector](#) &v)
- template<class InputIterator >  
  [Vector](#) (InputIterator first, InputIterator last)
- [Vector](#) ([Vector](#) &&v)
- [Vector](#) (const std::initializer\_list< T > il)
- [~Vector](#) ()  
*Destruktorius.*
- [Vector](#) & [operator=](#) (const [Vector](#) &other)  
*Operator = /copy assignment.*
- [Vector](#) & [operator=](#) ([Vector](#) &&other)
- 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
- 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 ()
- [const\\_iterator](#) begin () const
- [iterator](#) end ()
- [const\\_iterator](#) end () const
- [size\\_type](#) size () const
- [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
- [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`
- `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`)

### Private Member Functions

- `void create` ()
- `void create` (`size_type n`, `const T &val`)
- `void create` (`const_iterator i`, `const_iterator j`)
- `void uncreate` ()
- `void grow` (`size_type new_capacity=1`)
- `void unchecked_append` (`const T &val`)

### Private Attributes

- `iterator dat`
- `iterator avail`
- `iterator limit`
- `std::allocator< T > alloc`

## 4.2.1 Member Typedef Documentation

### 4.2.1.1 `const_iterator`

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

Konstantos iteratoriaus tipo narys.

### 4.2.1.2 `const_reference`

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

Konstantos nuorodos tipo narys.

#### 4.2.1.3 iterator

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

Iteratoriaus tipo narys.

#### 4.2.1.4 reference

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

Nuorodos tipo narys.

#### 4.2.1.5 size\_type

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

Dydžio tipo narys.

#### 4.2.1.6 value\_type

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

Reikšmės tipo narys.

### 4.2.2 Constructor & Destructor Documentation

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

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

Konstruktoriai /default.

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

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

#### 4.2.2.3 Vector() [3/6]

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

#### 4.2.2.4 Vector() [4/6]

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

#### 4.2.2.5 Vector() [5/6]

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

#### 4.2.2.6 Vector() [6/6]

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

#### 4.2.2.7 ~Vector()

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

Destruktoriüs.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 assign() [1/3]

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

Assign.

#### 4.2.3.2 assign() [2/3]

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

#### 4.2.3.3 assign() [3/3]

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

#### 4.2.3.4 at() [1/2]

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

#### 4.2.3.5 at() [2/2]

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

#### 4.2.3.6 back() [1/2]

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

#### 4.2.3.7 back() [2/2]

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

#### 4.2.3.8 begin() [1/2]

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

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

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



#### 4.2.3.10 capacity()

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

#### 4.2.3.11 clear()

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

#### 4.2.3.12 create() [1/3]

```
template<typename T >
void Vector< T >::create ( ) [inline], [private]
```

#### 4.2.3.13 create() [2/3]

```
template<typename T >
void Vector< T >::create (
    const_iterator i,
    const_iterator j ) [inline], [private]
```

#### 4.2.3.14 create() [3/3]

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

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

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

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

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

#### 4.2.3.17 empty()

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

**4.2.3.18 end()** [1/2]

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

**4.2.3.19 end()** [2/2]

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

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

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

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

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

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

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

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

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

**4.2.3.24 grow()**

```
template<typename T >
void Vector< T >::grow (
    size_type new_capacity = 1 ) [inline], [private]
```

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

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

#### 4.2.3.26 insert() [2/2]

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

#### 4.2.3.27 max\_size()

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

#### 4.2.3.28 operator!=(())

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

#### 4.2.3.29 operator<()

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

#### 4.2.3.30 operator<=()

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

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

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

Operator = /copy assignment.

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

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

#### 4.2.3.33 operator==()

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

#### 4.2.3.34 operator>()

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

#### 4.2.3.35 operator>=()

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

#### 4.2.3.36 operator[]() [1/2]

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

#### 4.2.3.37 operator[]() [2/2]

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

#### 4.2.3.38 pop\_back()

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

#### 4.2.3.39 push\_back() [1/2]

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

#### 4.2.3.40 push\_back() [2/2]

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

**4.2.3.41 reserve()**

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

**4.2.3.42 resize() [1/2]**

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

**4.2.3.43 resize() [2/2]**

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

**4.2.3.44 shrink\_to\_fit()**

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

**4.2.3.45 size()**

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

**4.2.3.46 swap() [1/2]**

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

**4.2.3.47 swap() [2/2]**

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

**4.2.3.48 unchecked\_append()**

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

#### 4.2.3.49 uncreate()

```
template<typename T >
void Vector< T >::uncreate ( ) [inline], [private]
```

### 4.2.4 Member Data Documentation

#### 4.2.4.1 alloc

```
template<typename T >
std::allocator<T> Vector< T >::alloc [private]
```

#### 4.2.4.2 avail

```
template<typename T >
iterator Vector< T >::avail [private]
```

#### 4.2.4.3 dat

```
template<typename T >
iterator Vector< T >::dat [private]
```

#### 4.2.4.4 limit

```
template<typename T >
iterator Vector< T >::limit [private]
```

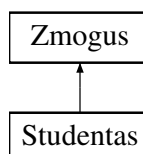
The documentation for this class was generated from the following file:

- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/[vector.h](#)

### 4.3 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\\_](#)

## 4.3.1 Constructor & Destructor Documentation

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

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

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

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

### 4.3.1.3 ~Zmogus()

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

## 4.3.2 Member Function Documentation

### 4.3.2.1 getPavarde()

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

Implemented in [Studentas](#).

### 4.3.2.2 getVardas()

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

Implemented in [Studentas](#).

#### 4.3.2.3 setPavarde()

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

Reimplemented in [Studentas](#).

#### 4.3.2.4 setVardas()

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

Reimplemented in [Studentas](#).

### 4.3.3 Member Data Documentation

#### 4.3.3.1 pavarde\_

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

#### 4.3.3.2 vardas\_

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

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

- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/[studentas.h](#)



# Chapter 5

## File Documentation

### 5.1 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/Funkcijos.cpp File Reference

```
#include "funkcijos.h"
#include "vector.h"
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <sstream>
#include <fstream>
#include <chrono>
#include <list>
#include <deque>
#include <cassert>
```

#### Functions

- double [mediana](#) ([Vector](#)< int > pazymiai, int egzamino\_rezultatas)
- double [vidurkis\\_galutinis](#) (double [suma](#), int [n](#), int egzamino\_rezultatas)
- bool [palyginti\\_pagal\\_varda](#) (const [Studentas](#) &a, const [Studentas](#) &b)
- bool [palyginti\\_pagal\\_pavarde](#) (const [Studentas](#) &a, const [Studentas](#) &b)
- bool [palyginti\\_pagal\\_mediana](#) (const [Studentas](#) &a, const [Studentas](#) &b)
- bool [palyginti\\_pagal\\_galutini\\_vidurki](#) (const [Studentas](#) &a, const [Studentas](#) &b)
- bool [palyginti\\_pagal\\_galutini\\_vidurki\\_didejimo\\_tvarka](#) (const [Studentas](#) &a, const [Studentas](#) &b)
- bool [ar\\_vargsiukas](#) (const [Studentas](#) &student)
- void [RezultatuVaizdavimas](#) (const [Vector](#)< [Studentas](#) > &studentai, int pasirinkimas1)
- void [generuotiFaila](#) (string failoPavadinimas, int ndSkaicius, int studentuSkaicius)
- void [failolsvedimasVector1](#) (const [Vector](#)< [Studentas](#) > &studentai, string failoPavadinimas)
- void [nuskaitytiFailaVector1](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailoPavadinimas)
- void [failolsvedimasList1](#) (const list< [Studentas](#) > &studentai, string failoPavadinimas)
- void [nuskaitytiFailaList1](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailoPavadinimas)
- void [failolsvedimasDeque1](#) (const deque< [Studentas](#) > &studentai, string failoPavadinimas)
- void [nuskaitytiFailaDeque1](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailoPavadinimas)

- void [nuskaitytiFailaVector2](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaList2](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaDeque2](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaVector3](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaList3](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaDeque3](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [testai](#) ()

## 5.1.1 Function Documentation

### 5.1.1.1 ar\_vargsiukas()

```
bool ar_vargsiukas (
    const Studentas & student )
```

### 5.1.1.2 failoIsvedimasDeque1()

```
void failoIsvedimasDeque1 (
    const deque< Studentas > & studentai,
    string failoPavadinimas )
```

### 5.1.1.3 failoIsvedimasList1()

```
void failoIsvedimasList1 (
    const list< Studentas > & studentai,
    string failoPavadinimas )
```

### 5.1.1.4 failoIsvedimasVector1()

```
void failoIsvedimasVector1 (
    const Vector< Studentas > & studentai,
    string failoPavadinimas )
```

### 5.1.1.5 generuotiFaila()

```
void generuotiFaila (
    string failoPavadinimas,
    int ndSkaicius,
    int studentuSkaicius )
```

### 5.1.1.6 mediana()

```
double mediana (
    Vector< int > pazymiai,
    int egzamino_rezultatas )
```

#### 5.1.1.7 nuskaitytiFailaDeque1()

```
void nuskaitytiFailaDeque1 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas,
    string kietakiuFailoPavadinimas )
```

#### 5.1.1.8 nuskaitytiFailaDeque2()

```
void nuskaitytiFailaDeque2 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.1.1.9 nuskaitytiFailaDeque3()

```
void nuskaitytiFailaDeque3 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.1.1.10 nuskaitytiFailaList1()

```
void nuskaitytiFailaList1 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas,
    string kietakiuFailoPavadinimas )
```

#### 5.1.1.11 nuskaitytiFailaList2()

```
void nuskaitytiFailaList2 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.1.1.12 nuskaitytiFailaList3()

```
void nuskaitytiFailaList3 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.1.1.13 nuskaitytiFailaVector1()

```
void nuskaitytiFailaVector1 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas,
    string kietakiuFailoPavadinimas )
```

#### 5.1.1.14 nuskaitytiFailaVector2()

```
void nuskaitytiFailaVector2 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.1.1.15 nuskaitytiFailaVector3()

```
void nuskaitytiFailaVector3 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.1.1.16 palyginti\_pagal\_galutini\_vidurki()

```
bool palyginti_pagal_galutini_vidurki (
    const Studentas & a,
    const Studentas & b )
```

#### 5.1.1.17 palyginti\_pagal\_galutini\_vidurki\_didejimo\_tvarka()

```
bool palyginti_pagal_galutini_vidurki_didejimo_tvarka (
    const Studentas & a,
    const Studentas & b )
```

#### 5.1.1.18 palyginti\_pagal\_mediana()

```
bool palyginti_pagal_mediana (
    const Studentas & a,
    const Studentas & b )
```

#### 5.1.1.19 palyginti\_pagal\_pavarde()

```
bool palyginti_pagal_pavarde (
    const Studentas & a,
    const Studentas & b )
```

#### 5.1.1.20 palyginti\_pagal\_varda()

```
bool palyginti_pagal_varda (
    const Studentas & a,
    const Studentas & b )
```

#### 5.1.1.21 RezultatuVaizdavimas()

```
void RezultatuVaizdavimas (
    const Vector< Studentas > & studentai,
    int pasirinkimas1 )
```

### 5.1.1.22 testai()

```
void testai ( )
```

### 5.1.1.23 vidurkis\_galutinis()

```
double vidurkis_galutinis (
    double suma,
    int n,
    int egzamino_rezultatas )
```

## 5.2 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/funkcijos.h File Reference

```
#include <vector>
#include <list>
#include <deque>
#include "studentas.h"
#include "vector.h"
```

### Functions

- double [mediana](#) (Vector< int > pazymiai, int egzamino\_rezultatas)
- double [vidurkis\\_galutinis](#) (double suma, int n, int egzamino\_rezultatas)
- bool [palyginti\\_pagal\\_varda](#) (const Studentas &a, const Studentas &b)
- bool [palyginti\\_pagal\\_pavarde](#) (const Studentas &a, const Studentas &b)
- bool [palyginti\\_pagal\\_mediana](#) (const Studentas &a, const Studentas &b)
- bool [palyginti\\_pagal\\_galutini\\_vidurki](#) (const Studentas &a, const Studentas &b)
- bool [palyginti\\_pagal\\_galutini\\_vidurki\\_didejimo\\_tvarka](#) (const Studentas &a, const Studentas &b)
- bool [ar\\_vargsiukas](#) (const Studentas &student)
- void [RezultatuVaizdavimas](#) (const Vector< Studentas > &studentai, int pasirinkimas1)
- void [generuotiFaila](#) (string failoPavadinimas, int ndSkaicius, int studentuSkaicius)
- void [failoIsvedimasVector1](#) (const Vector< Studentas > &studentai, string failoPavadinimas)
- void [nuskaitytiFailaVector1](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailoPavadinimas)
- void [failoIsvedimasList1](#) (const list< Studentas > &studentai, string failoPavadinimas)
- void [nuskaitytiFailaList1](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailoPavadinimas)
- void [failoIsvedimasDeque1](#) (const deque< Studentas > &studentai, string failoPavadinimas)
- void [nuskaitytiFailaDeque1](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailoPavadinimas)
- void [nuskaitytiFailaVector2](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaList2](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaDeque2](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaVector3](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaList3](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [nuskaitytiFailaDeque3](#) (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void [testai](#) ()

## 5.2.1 Function Documentation

### 5.2.1.1 ar\_vargsiukas()

```
bool ar_vargsiukas (
    const Studentas & student )
```

### 5.2.1.2 failoIsvedimasDeque1()

```
void failoIsvedimasDeque1 (
    const deque< Studentas > & studentai,
    string failoPavadinimas )
```

### 5.2.1.3 failoIsvedimasList1()

```
void failoIsvedimasList1 (
    const list< Studentas > & studentai,
    string failoPavadinimas )
```

### 5.2.1.4 failoIsvedimasVector1()

```
void failoIsvedimasVector1 (
    const Vector< Studentas > & studentai,
    string failoPavadinimas )
```

### 5.2.1.5 generuotiFaila()

```
void generuotiFaila (
    string failoPavadinimas,
    int ndSkaicius,
    int studentuSkaicius )
```

### 5.2.1.6 mediana()

```
double mediana (
    Vector< int > pazymiai,
    int egzamino_rezultatas )
```

### 5.2.1.7 nuskaitytiFailaDeque1()

```
void nuskaitytiFailaDeque1 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas,
    string kietakiuFailoPavadinimas )
```

#### 5.2.1.8 nuskaitytiFailaDeque2()

```
void nuskaitytiFailaDeque2 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.2.1.9 nuskaitytiFailaDeque3()

```
void nuskaitytiFailaDeque3 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.2.1.10 nuskaitytiFailaList1()

```
void nuskaitytiFailaList1 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas,
    string kietakiuFailoPavadinimas )
```

#### 5.2.1.11 nuskaitytiFailaList2()

```
void nuskaitytiFailaList2 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.2.1.12 nuskaitytiFailaList3()

```
void nuskaitytiFailaList3 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.2.1.13 nuskaitytiFailaVector1()

```
void nuskaitytiFailaVector1 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas,
    string kietakiuFailoPavadinimas )
```

#### 5.2.1.14 nuskaitytiFailaVector2()

```
void nuskaitytiFailaVector2 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.2.1.15 nuskaitytiFailaVector3()

```
void nuskaitytiFailaVector3 (
    string failoPavadinimas,
    string vargsiukuFailoPavadinimas )
```

#### 5.2.1.16 palyginti\_pagal\_galutini\_vidurki()

```
bool palyginti_pagal_galutini_vidurki (
    const Studentas & a,
    const Studentas & b )
```

#### 5.2.1.17 palyginti\_pagal\_galutini\_vidurki\_didejimo\_tvarka()

```
bool palyginti_pagal_galutini_vidurki_didejimo_tvarka (
    const Studentas & a,
    const Studentas & b )
```

#### 5.2.1.18 palyginti\_pagal\_mediana()

```
bool palyginti_pagal_mediana (
    const Studentas & a,
    const Studentas & b )
```

#### 5.2.1.19 palyginti\_pagal\_pavarde()

```
bool palyginti_pagal_pavarde (
    const Studentas & a,
    const Studentas & b )
```

#### 5.2.1.20 palyginti\_pagal\_varda()

```
bool palyginti_pagal_varda (
    const Studentas & a,
    const Studentas & b )
```

#### 5.2.1.21 RezultatuVaizdavimas()

```
void RezultatuVaizdavimas (
    const Vector< Studentas > & studentai,
    int pasirinkimas1 )
```

#### 5.2.1.22 testai()

```
void testai ( )
```



## 5.2.1.23 vidurkis\_galutinis()

```
double vidurkis_galutinis (
    double suma,
    int n,
    int egzamino_rezultatas )
```

## 5.3 funkcijos.h

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

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include <vector>
00005 #include <list>
00006 #include <deque>
00007 #include "studentas.h"
00008 #include "vector.h"
00009
00010 using namespace std;
00011
00012 double mediana(Vector<int> pazymiai, int egzamino_rezultatas);
00013 double vidurkis_galutinis(double suma, int n, int egzamino_rezultatas);
00014 bool palyginti_pagal_varda(const Studentas &a, const Studentas &b);
00015 bool palyginti_pagal_pavarde(const Studentas &a, const Studentas &b);
00016 bool palyginti_pagal_mediana(const Studentas &a, const Studentas &b);
00017 bool palyginti_pagal_galutini_vidurki(const Studentas &a, const Studentas &b);
00018 bool palyginti_pagal_galutini_vidurki_didejimo_tvarka(const Studentas &a, const Studentas &b);
00019 bool ar_vargsiukas(const Studentas& student);
00020 void RezultatuVaizdavimas (const Vector<Studentas>& studentai, int pasirinkimas1);
00021 void generuotiFaila(string failoPavadinimas, int ndSkaicius, int studentuSkaicius);
00022
00023 // 1 strategija
00024 void failoIsvedimasVector1(const Vector<Studentas>& studentai, string failoPavadinimas);
00025 void nuskaitytiFailaVector1(string failoPavadinimas, string vargsiukuFailoPavadinimas, string
kietakiuFailoPavadinimas);
00026 void failoIsvedimasList1(const list<Studentas>& studentai, string failoPavadinimas);
00027 void nuskaitytiFailaList1(string failoPavadinimas, string vargsiukuFailoPavadinimas, string
kietakiuFailoPavadinimas);
00028 void failoIsvedimasDeque1(const deque<Studentas>& studentai, string failoPavadinimas);
00029 void nuskaitytiFailaDeque1(string failoPavadinimas, string vargsiukuFailoPavadinimas, string
kietakiuFailoPavadinimas);
00030
00031 // 2 strategija
00032 void nuskaitytiFailaVector2(string failoPavadinimas, string vargsiukuFailoPavadinimas);
00033 void nuskaitytiFailaList2(string failoPavadinimas, string vargsiukuFailoPavadinimas);
00034 void nuskaitytiFailaDeque2(string failoPavadinimas, string vargsiukuFailoPavadinimas);
00035
00036 // 3 strategija
00037 void nuskaitytiFailaVector3(string failoPavadinimas, string vargsiukuFailoPavadinimas);
00038 void nuskaitytiFailaList3(string failoPavadinimas, string vargsiukuFailoPavadinimas);
00039 void nuskaitytiFailaDeque3(string failoPavadinimas, string vargsiukuFailoPavadinimas);
00040
00041 void testai();
00042
00043 #endif
```

## 5.4 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/studentas.h File Reference

```
#include "vector.h"
#include <string>
#include <iostream>
```

## Classes

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

## 5.5 studentas.h

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

```

00001 #ifndef STUDENTAS_H
00002 #define STUDENTAS_H
00003
00004 #include "vector.h"
00005 #include <string>
00006 #include <iostream>
00007
00008 using namespace std;
00009
00010 class Zmogus {
00011 protected:
00012     string vardas_;
00013     string pavarde_;
00014
00015 public:
00016
00017     Zmogus() = default;
00018     Zmogus(const string& vardas, const string& pavarde) : vardas_(vardas), pavarde_(pavarde) {}
00019     virtual ~Zmogus() {}
00020
00021     virtual void setVardas(const string& vardas) { vardas_ = vardas; }
00022     virtual void setPavarde(const string& pavarde) { pavarde_ = pavarde; }
00023     virtual string getVardas() const = 0; // { return vardas_; }
00024     virtual string getPavarde() const = 0; // { return pavarde_; }
00025 };
00026
00027 class Studentas : public Zmogus {
00028 private:
00029     Vector<int> pazymiai_;
00030     int egzamino_rezultatas_;
00031     double galutinis_vid_, mediana_;
00032
00033 public:
00034     Studentas() : egzamino_rezultatas_(0), galutinis_vid_(0), mediana_(0) {}
00035     Studentas(const Vector<int>& pazymiai, const string& vardas, const string& pavarde, int
00036 egzamino_rezultatas, double galutinis_vid, double mediana)
00037         : Zmogus(vardas, pavarde), pazymiai_(pazymiai), egzamino_rezultatas_(egzamino_rezultatas),
00038 galutinis_vid_(galutinis_vid), mediana_(mediana) {}
00039
00040     // Copy constructor
00041     Studentas(const Studentas& other)
00042         : Zmogus(other.getVardas(), other.getPavarde()), pazymiai_(other.pazymiai_),
00043 egzamino_rezultatas_(other.egzamino_rezultatas_),
00044 galutinis_vid_(other.galutinis_vid_), mediana_(other.mediana_) {}
00045
00046     // Copy assignment
00047     Studentas& operator = (const Studentas& other)
00048     {
00049         if (this != &other) {
00050             Zmogus::setVardas(other.getVardas());
00051             Zmogus::setPavarde(other.getPavarde());
00052             pazymiai_ = other.pazymiai_;
00053             egzamino_rezultatas_ = other.egzamino_rezultatas_;
00054             galutinis_vid_ = other.galutinis_vid_;
00055             mediana_ = other.mediana_;
00056         }
00057         return *this;
00058     }
00059
00060     // Move constructor
00061     Studentas(Studentas&& other) noexcept
00062         : Zmogus(move(other.vardas_), move(other.pavarde_)), pazymiai_(move(other.pazymiai_)),
00063 egzamino_rezultatas_(other.egzamino_rezultatas_),
00064 galutinis_vid_(other.galutinis_vid_), mediana_(other.mediana_) {other.egzamino_rezultatas_ =
00065 0; other.galutinis_vid_ = 0;
00066 other.mediana_ = 0; }
00067
00068     // Move assignment
00069     Studentas& operator = (Studentas&& other) noexcept
00070     {
00071         if (this != &other) {
00072             Zmogus::setVardas(move(other.getVardas()));
00073             Zmogus::setPavarde(move(other.getPavarde()));
00074             pazymiai_ = move(other.pazymiai_);
00075             egzamino_rezultatas_ = other.egzamino_rezultatas_;
00076             galutinis_vid_ = other.galutinis_vid_;
00077             mediana_ = other.mediana_;
00078             other.egzamino_rezultatas_ = 0;
00079             other.galutinis_vid_ = 0;
00080             other.mediana_ = 0;
00081         }
00082         return *this;
00083     }

```

```

00078     }
00079
00080     ~Studentas() { pazymiai_.clear(); }
00081
00082     void setVardas(const string& vardas) override { Zmogus::setVardas(vardas); }
00083     void setPavarde(const string& pavarde) override { Zmogus::setPavarde(pavarde); }
00084     void setPazymiai(const Vector<int>& pazymiai) { pazymiai_ = pazymiai; }
00085     void setEgzaminoRezultatas(int egzamino_rezultatas) { egzamino_rezultatas_ = egzamino_rezultatas; }
00086 }
00087
00088     void setGalutinisVid(double galutinis_vid) { galutinis_vid_ = galutinis_vid; }
00089     void setMediana(double mediana) { mediana_ = mediana; }
00090
00091     string getVardas() const override { return vardas_; }
00092     string getPavarde() const override { return pavarde_; }
00093     const Vector<int>& getPazymiai() const { return pazymiai_; }
00094     int getEgzaminoRezultatas() const { return egzamino_rezultatas_; }
00095     double getGalutinisVid() const { return galutinis_vid_; }
00096     double getMediana() const { return mediana_; }
00097
00098     void addPazymys(int pazymys) { pazymiai_.push_back(pazymys); }
00099
00100     // Input
00101     friend istream& operator>(istream& is, Studentas& studentas) {
00102         is >> studentas.vardas_ >> studentas.pavarde_;
00103
00104         int pazymys;
00105         studentas.pazymiai_.clear();
00106         while(is >> pazymys && pazymys > 0 && pazymys <= 10 ) {
00107             studentas.addPazymys(pazymys);
00108         }
00109
00110         is >> studentas.egzamino_rezultatas_;
00111
00112         return is;
00113     }
00114
00115     // Output
00116     friend ostream& operator<<(ostream& os, const Studentas& studentas) {
00117         os << "Studento informacija:" << endl;
00118         os << "Vardas: " << studentas.vardas_ << endl;
00119         os << "Pavarde: " << studentas.pavarde_ << endl;
00120         os << "Galutinis vidurkis: " << studentas.galutinis_vid_ << endl;
00121         return os;
00122     }
00123 };
00124
00125 #endif

```

## 5.6 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/vector.h File Reference

```

#include <iostream>
#include <memory>
#include <algorithm>
#include <limits>

```

### Classes

- class [Vector< T >](#)

## 5.7 vector.h

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

```

00001 #pragma once
00002

```

```

00003 #include <iostream>
00004 #include <memory>
00005 #include <algorithm>
00006 #include <limits>
00007
00008 template <typename T>
00009 class Vector{
00010     public:
00011
00012     // MEMBER TYPE
00013     typedef size_t size_type;
00014     typedef T value_type;
00015     typedef T& reference;
00016     typedef const T& const_reference;
00017     typedef T* iterator;
00018     typedef const T* const_iterator;
00020     // MEMBER FUNCTIONS
00021
00024     Vector() {create();}
00026     explicit Vector(size_type n, const T& t = T{}) { create (n,t); }
00028     Vector(const Vector& v) { create(v.begin(), v.end()); }
00030     template <class InputIterator>
00031     Vector (InputIterator first, InputIterator last) { create(first,last); }
00033     Vector (Vector&& v) {
00034         create();
00035         swap(v);
00036         v.uncreate();
00037     }
00039     Vector(const std::initializer_list<T> il) { create(il.begin(), il.end()); }
00040
00042     ~Vector() {uncreate();}
00043
00046     Vector& operator = (const Vector& other) {
00047         if (this != &other) {
00048             uncreate();
00049             create(other.begin(), other.end());
00050         }
00051         return *this;
00052     };
00053
00055     Vector& operator = (Vector&& other) {
00056         if (this != &other) {
00057             create(other.begin(), other.end());
00058             uncreate();
00059         }
00060         return *this;
00061     }
00062
00064     template <class InputIterator>
00065     void assign (InputIterator first, InputIterator last) {
00066         uncreate();
00067         create(first, last);
00068     }
00069     void assign (size_type n, const value_type& val) {
00070         uncreate();
00071         create(n, val);
00072     }
00073     void assign (std::initializer_list<value_type> il) {
00074         uncreate();
00075         create(il);
00076     }
00077
00078     // Element access
00079     const_reference at (size_type n) const {
00080         if (n >= size() || n < 0)
00081             throw std::out_of_range("Index out of range");
00082         return dat[n];
00083     }
00084
00085     T& operator[] (size_type n) {return dat[n];}
00086     const T& operator[] (size_type n) const {return dat[n];}
00087     reference at (size_type n) {
00088         if (n >= size() || n < 0)
00089             throw std::out_of_range("Index out of range");
00090         return dat[n];
00091     }
00092
00093     reference front() {
00094         return dat[0];
00095     };
00096
00097     const_reference front() const {
00098         return dat[0];
00099     }
00100
00101     reference back() {
00102         return dat[size() - 1];

```

```

00103     }
00104
00105     const_reference back() const {
00106         return dat[size() - 1];
00107     }
00108
00109     value_type* data() noexcept {
00110         return dat;
00111     }
00112
00113     const value_type* data() const noexcept {
00114         return dat;
00115     }
00116
00117 // Iterators
00118     iterator begin() {return dat;}
00119     const_iterator begin() const {return dat;}
00120     iterator end() {return avail;}
00121     const_iterator end() const {return avail;}
00122
00123 // Capacity
00124     size_type size() const {return avail-dat;}
00125     size_type max_size() const {return std::numeric_limits<size_type>::max();}
00126     void resize(size_type sz) {
00127         if (sz < size()) {
00128             iterator it = dat + sz;
00129             while (it != avail) {
00130                 alloc.destroy(it++);
00131             }
00132             avail = dat + sz;
00133         }
00134         else if (sz > capacity()) {
00135             grow(sz);
00136             std::uninitialized_fill(avail, dat + sz, value_type());
00137             avail = dat + sz;
00138         }
00139         else if (sz > size()) {
00140             std::uninitialized_fill(avail, dat + sz, value_type());
00141             avail = dat + sz;
00142         }
00143     }
00144
00145     void resize(size_type sz, const value_type& value) {
00146         if (sz > capacity()) {
00147             grow(sz);
00148         }
00149
00150         if (sz > size()) {
00151             insert(end(), sz - size(), value);
00152         } else if (sz < size()) {
00153             avail = dat + sz;
00154         }
00155     }
00156
00157     size_type capacity() const {return limit-dat;}
00158     bool empty() const noexcept { return size() == 0;}
00159     void reserve (size_type n) {
00160         if (n > capacity()) {
00161             grow(n);
00162         }
00163     }
00164     void shrink_to_fit(){
00165         if (limit > avail)
00166             limit = avail;
00167     }
00168
00169 // Modifiers
00170     void clear() noexcept {
00171         uncreate();
00172     }
00173
00174     iterator insert (const_iterator position, const value_type& val) {
00175         return insert(position, 1, val);
00176     }
00177     iterator insert(iterator position, size_type n, const value_type& val) {
00178         if (position < begin() || position > end()) {
00179             throw std::out_of_range("Index out of range");
00180         }
00181         if (avail + n > limit) {
00182             size_type index = position - begin();
00183             grow(n);
00184             position = begin() + index;
00185         }
00186         for (iterator it = end() + n - 1; it != position + n - 1; --it) {
00187             *it = std::move(*(it - n));
00188         }
00189         std::uninitialized_fill(position, position + n, val);

```

```

00190         avail += n;
00191
00192         return position;
00193     }
00194
00195     iterator erase(iterator position) {
00196         if (position < dat || position > avail) {
00197             throw std::out_of_range("Index out of range");
00198         }
00199         std::move(position + 1, avail, position);
00200         alloc.destroy(avail - 1);
00201         --avail;
00202
00203         return position;
00204     }
00205
00206     iterator erase(iterator first, iterator last) {
00207         iterator new_available = std::uninitialized_copy(last, avail, first);
00208
00209         iterator it = avail;
00210         while (it != new_available) {
00211             alloc.destroy(--it);
00212         }
00213
00214         avail = new_available;
00215         return last;
00216     }
00217
00218     void push_back (const value_type& t) {
00219         if (avail==limit)
00220             grow();
00221         unchecked_append(t);
00222     }
00223
00224     void push_back (value_type&& val) {
00225         if (avail == limit)
00226             grow();
00227         unchecked_append(val);
00228     }
00229
00230     void pop_back() {
00231         if (avail != dat)
00232             alloc.destroy(--avail);
00233     }
00234
00235     void swap(Vector& x) {
00236         std::swap(dat, x.dat);
00237         std::swap(avail, x.avail);
00238         std::swap(limit, x.limit);
00239     }
00240
00241
00242     // NON-MEMBER FUNCTIONS
00243
00244     bool operator== (const Vector<T>& other) const {
00245         if (size() != other.size()) {
00246             return false;
00247         }
00248
00249         return std::equal(begin(), end(), other.begin());
00250     }
00251     bool operator!= (const Vector<T>& other) const {
00252         return !(*this == other);
00253     }
00254     bool operator < (const Vector<T> & other) const {
00255         return std::lexicographical_compare(begin(), end(), other.begin(), other.end());
00256     }
00257     bool operator <= (const Vector<T> & other) const {
00258         return !(other < *this);
00259     }
00260     bool operator > (const Vector<T> & other) const {
00261         return std::lexicographical_compare(other.begin(), other.end(), begin(), end());
00262     }
00263     bool operator >= (const Vector<T> & other) const {
00264         return !(other > *this);
00265     }
00266
00267     void swap (Vector<T>& x, Vector<T>& y) {
00268         std::swap(x,y);
00269     }
00270
00271 private:
00272     iterator dat;
00273     iterator avail;
00274     iterator limit;
00275     std::allocator<T> alloc;
00276     void create() {dat = avail = limit = nullptr;}

```

```

00277     void create (size_type n, const T& val) {
00278         dat = alloc.allocate(n);
00279         limit = avail = dat + n;
00280         std::uninitialized_fill(dat, limit, val);
00281     }
00282     void create(const_iterator i, const_iterator j) {
00283         dat = alloc.allocate(j - i);
00284         limit = avail = std::uninitialized_copy(i, j, dat);
00285     }
00286     void uncreate() {
00287         if (dat) {
00288             iterator it = avail;
00289             while (it != dat) {
00290                 alloc.destroy(--it);
00291             }
00292             alloc.deallocate(dat, limit - dat);
00293         }
00294         dat = limit = avail = nullptr;
00295     }
00296     void grow(size_type new_capacity = 1) {
00297         size_type new_size = std::max(new_capacity, 2 * capacity());
00298         iterator new_data = alloc.allocate(new_size);
00299         iterator new_avail = std::uninitialized_copy(dat, avail, new_data);
00300         uncreate();
00301         dat = new_data;
00302         avail = new_avail;
00303         limit = dat + new_size;
00304     }
00305     void unchecked_append(const T& val) {
00306         alloc.construct(avail++, val);
00307     }
00308 };

```

## 5.8 C:/Users/acer/Desktop/Adri/VSCode/Programavimas/Vektoriai.cpp File Reference

```

#include <iostream>
#include <cmath>
#include <math.h>
#include <iomanip>
#include <string>
#include <algorithm>
#include <sstream>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <fstream>
#include <chrono>
#include <random>
#include "studentas.h"
#include "funkcijos.h"
#include "vector.h"

```

### Functions

- int [main](#) ()

### Variables

- int [m](#)
- int [n](#)
- int [menu](#)

- int `variantas_namu_darbas`
- int `variantas_studentas`
- int `pasirinkimas`
- double `suma`
- double `vidurkis`
- int `MAX_ND` = 30
- int `MAX_STUDENTU` = 30
- string `raide`

## 5.8.1 Function Documentation

### 5.8.1.1 `main()`

```
int main ( )
```

## 5.8.2 Variable Documentation

### 5.8.2.1 `m`

```
int m
```

### 5.8.2.2 `MAX_ND`

```
int MAX_ND = 30
```

### 5.8.2.3 `MAX_STUDENTU`

```
int MAX_STUDENTU = 30
```

### 5.8.2.4 `menu`

```
int menu
```

### 5.8.2.5 `n`

```
int n
```

### 5.8.2.6 `pasirinkimas`

```
int pasirinkimas
```



#### 5.8.2.7 raide

```
string raide
```

#### 5.8.2.8 suma

```
double suma
```

#### 5.8.2.9 variantas\_namu\_darbas

```
int variantas_namu_darbas
```

#### 5.8.2.10 variantas\_studentas

```
int variantas_studentas
```

#### 5.8.2.11 vidurkis

```
double vidurkis
```



# Index

- ~Studentas
  - Studentas, [9](#)
- ~Vector
  - Vector< T >, [15](#)
- ~Zmogus
  - Zmogus, [23](#)
- addPazymys
  - Studentas, [9](#)
- alloc
  - Vector< T >, [22](#)
- ar\_vargsiukas
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- assign
  - Vector< T >, [15](#), [16](#)
- at
  - Vector< T >, [16](#)
- avail
  - Vector< T >, [22](#)
- back
  - Vector< T >, [16](#)
- begin
  - Vector< T >, [16](#)
- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/Funkcijos.cpp,
  - [25](#)
- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/funkcijos.h,
  - [29](#), [33](#)
- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/studentas.h,
  - [33](#), [34](#)
- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/vector.h,
  - [35](#)
- C:/Users/acer/Desktop/Adri/VSCode/Programavimas/Vektoriai.cpp,
  - [39](#)
- capacity
  - Vector< T >, [16](#)
- clear
  - Vector< T >, [17](#)
- const\_iterator
  - Vector< T >, [13](#)
- const\_reference
  - Vector< T >, [13](#)
- create
  - Vector< T >, [17](#)
- dat
  - Vector< T >, [22](#)
- data
  - Vector< T >, [17](#)
- egzamino\_rezultatas\_
  - Studentas, [11](#)
- empty
  - Vector< T >, [17](#)
- end
  - Vector< T >, [17](#), [18](#)
- erase
  - Vector< T >, [18](#)
- failolsvedimasDeque1
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- failolsvedimasList1
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- failolsvedimasVector1
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- front
  - Vector< T >, [18](#)
- Funkcijos.cpp
  - ar\_vargsiukas, [26](#)
  - failolsvedimasDeque1, [26](#)
  - failolsvedimasList1, [26](#)
  - failolsvedimasVector1, [26](#)
  - generuotiFaila, [26](#)
  - mediana, [26](#)
  - nuskaitytiFailaDeque1, [26](#)
  - nuskaitytiFailaDeque2, [27](#)
  - nuskaitytiFailaDeque3, [27](#)
  - nuskaitytiFailaList1, [27](#)
  - nuskaitytiFailaList2, [27](#)
  - nuskaitytiFailaList3, [27](#)
  - nuskaitytiFailaVector1, [27](#)
  - nuskaitytiFailaVector2, [27](#)
  - nuskaitytiFailaVector3, [28](#)
  - palyginti\_pagal\_galutini\_vidurki, [28](#)
  - palyginti\_pagal\_galutini\_vidurki\_didejimo\_tvarka, [28](#)
  - palyginti\_pagal\_mediana, [28](#)
  - palyginti\_pagal\_pavarde, [28](#)
  - palyginti\_pagal\_varda, [28](#)
  - RezultatuVaizdavimas, [28](#)
  - testai, [28](#)
  - vidurkis\_galutinis, [29](#)
- funkcijos.h
  - ar\_vargsiukas, [30](#)
  - failolsvedimasDeque1, [30](#)

- failolsvedimasList1, [30](#)
- failolsvedimasVector1, [30](#)
- generuotiFaila, [30](#)
- mediana, [30](#)
- nuskaitytiFailaDeque1, [30](#)
- nuskaitytiFailaDeque2, [30](#)
- nuskaitytiFailaDeque3, [31](#)
- nuskaitytiFailaList1, [31](#)
- nuskaitytiFailaList2, [31](#)
- nuskaitytiFailaList3, [31](#)
- nuskaitytiFailaVector1, [31](#)
- nuskaitytiFailaVector2, [31](#)
- nuskaitytiFailaVector3, [31](#)
- palyginti\_pagal\_galutini\_vidurki, [32](#)
- palyginti\_pagal\_galutini\_vidurki\_didejimo\_tvarka, [32](#)
- palyginti\_pagal\_mediana, [32](#)
- palyginti\_pagal\_pavarde, [32](#)
- palyginti\_pagal\_varda, [32](#)
- RezultatuVaizdavimas, [32](#)
- testai, [32](#)
- vidurkis\_galutinis, [32](#)
- galutinis\_vid\_
  - Studentas, [11](#)
- generuotiFaila
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- getEgzaminoRezultatas
  - Studentas, [9](#)
- getGalutinisVid
  - Studentas, [9](#)
- getMediana
  - Studentas, [9](#)
- getPavarde
  - Studentas, [9](#)
  - Zmogus, [23](#)
- getPazymiai
  - Studentas, [9](#)
- getVardas
  - Studentas, [9](#)
  - Zmogus, [23](#)
- grow
  - Vector< T >, [18](#)
- insert
  - Vector< T >, [18](#)
- iterator
  - Vector< T >, [13](#)
- limit
  - Vector< T >, [22](#)
- m
  - Vektoriai.cpp, [40](#)
- main
  - Vektoriai.cpp, [40](#)
- MAX\_ND
  - Vektoriai.cpp, [40](#)
- max\_size
  - Vector< T >, [19](#)
- MAX\_STUDENTU
  - Vektoriai.cpp, [40](#)
- mediana
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- mediana\_
  - Studentas, [11](#)
- menu
  - Vektoriai.cpp, [40](#)
- n
  - Vektoriai.cpp, [40](#)
- nuskaitytiFailaDeque1
  - Funkcijos.cpp, [26](#)
  - funkcijos.h, [30](#)
- nuskaitytiFailaDeque2
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [30](#)
- nuskaitytiFailaDeque3
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- nuskaitytiFailaList1
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- nuskaitytiFailaList2
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- nuskaitytiFailaList3
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- nuskaitytiFailaVector1
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- nuskaitytiFailaVector2
  - Funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- nuskaitytiFailaVector3
  - Funkcijos.cpp, [28](#)
  - funkcijos.h, [31](#)
- operator!=
  - Vector< T >, [19](#)
- operator<
  - Vector< T >, [19](#)
- operator<<
  - Studentas, [11](#)
- operator<=
  - Vector< T >, [19](#)
- operator>
  - Vector< T >, [20](#)
- operator>>
  - Studentas, [11](#)
- operator>=
  - Vector< T >, [20](#)
- operator=
  - Studentas, [9](#), [10](#)
  - Vector< T >, [19](#)

- operator==
  - Vector< T >, 19
- operator[]
  - Vector< T >, 20
- palyginti\_pagal\_galutini\_vidurki
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- palyginti\_pagal\_galutini\_vidurki\_didejimo\_tvarka
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- palyginti\_pagal\_mediana
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- palyginti\_pagal\_pavarde
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- palyginti\_pagal\_varda
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- pasirinkimas
  - Vektoriai.cpp, 40
- pavarde\_
  - Zmogus, 24
- pazymiai\_
  - Studentas, 11
- pop\_back
  - Vector< T >, 20
- push\_back
  - Vector< T >, 20
- raide
  - Vektoriai.cpp, 40
- reference
  - Vector< T >, 14
- reserve
  - Vector< T >, 20
- resize
  - Vector< T >, 21
- RezultatuVaizdavimas
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- setEgzaminoRezultatas
  - Studentas, 10
- setGalutinisVid
  - Studentas, 10
- setMediana
  - Studentas, 10
- setPavarde
  - Studentas, 10
  - Zmogus, 23
- setPazymiai
  - Studentas, 10
- setVardas
  - Studentas, 10
  - Zmogus, 24
- shrink\_to\_fit
  - Vector< T >, 21
- size
  - Vector< T >, 21
- size\_type
  - Vector< T >, 14
- Studentas, 7
  - ~Studentas, 9
  - addPazymys, 9
  - egzamino\_rezultatas\_, 11
  - galutinis\_vid\_, 11
  - getEgzaminoRezultatas, 9
  - getGalutinisVid, 9
  - getMediana, 9
  - getPavarde, 9
  - getPazymiai, 9
  - getVardas, 9
  - mediana\_, 11
  - operator<<, 11
  - operator>>, 11
  - operator=, 9, 10
  - pazymiai\_, 11
  - setEgzaminoRezultatas, 10
  - setGalutinisVid, 10
  - setMediana, 10
  - setPavarde, 10
  - setPazymiai, 10
  - setVardas, 10
  - Studentas, 8
- suma
  - Vektoriai.cpp, 41
- swap
  - Vector< T >, 21
- testai
  - Funkcijos.cpp, 28
  - funkcijos.h, 32
- unchecked\_append
  - Vector< T >, 21
- uncreate
  - Vector< T >, 21
- value\_type
  - Vector< T >, 14
- vardas\_
  - Zmogus, 24
- variantas\_namu\_darbas
  - Vektoriai.cpp, 41
- variantas\_studentas
  - Vektoriai.cpp, 41
- Vector
  - Vector< T >, 14, 15
- Vector< T >, 11
  - ~Vector, 15
  - alloc, 22
  - assign, 15, 16
  - at, 16
  - avail, 22
  - back, 16
  - begin, 16

- capacity, [16](#)
- clear, [17](#)
- const\_iterator, [13](#)
- const\_reference, [13](#)
- create, [17](#)
- dat, [22](#)
- data, [17](#)
- empty, [17](#)
- end, [17](#), [18](#)
- erase, [18](#)
- front, [18](#)
- grow, [18](#)
- insert, [18](#)
- iterator, [13](#)
- limit, [22](#)
- max\_size, [19](#)
- operator!=, [19](#)
- operator<, [19](#)
- operator<=, [19](#)
- operator>, [20](#)
- operator>=, [20](#)
- operator=, [19](#)
- operator==, [19](#)
- operator[], [20](#)
- pop\_back, [20](#)
- push\_back, [20](#)
- reference, [14](#)
- reserve, [20](#)
- resize, [21](#)
- shrink\_to\_fit, [21](#)
- size, [21](#)
- size\_type, [14](#)
- swap, [21](#)
- unchecked\_append, [21](#)
- uncreate, [21](#)
- value\_type, [14](#)
- Vector, [14](#), [15](#)
- Vektoriai.cpp
  - m, [40](#)
  - main, [40](#)
  - MAX\_ND, [40](#)
  - MAX\_STUDENTU, [40](#)
  - menu, [40](#)
  - n, [40](#)
  - pasirinkimas, [40](#)
  - raide, [40](#)
  - suma, [41](#)
  - variantas\_namu\_darbas, [41](#)
  - variantas\_students, [41](#)
  - vidurkis, [41](#)
- vidurkis
  - Vektoriai.cpp, [41](#)
- vidurkis\_galutinis
  - Funkcijos.cpp, [29](#)
  - funkcijos.h, [32](#)
- Zmogus, [22](#)
  - ~Zmogus, [23](#)
  - getPavarde, [23](#)
  - getVardas, [23](#)
  - pavarde\_, [24](#)
  - setPavarde, [23](#)
  - setVardas, [24](#)
  - vardas\_, [24](#)
  - Zmogus, [23](#)