

v3.0 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 Vartotojas Class Reference	7
4.1.1 Detailed Description	9
4.1.2 Constructor & Destructor Documentation	9
4.1.2.1 Vartotojas() [1/4]	9
4.1.2.2 Vartotojas() [2/4]	9
4.1.2.3 ~Vartotojas()	9
4.1.2.4 Vartotojas() [3/4]	9
4.1.2.5 Vartotojas() [4/4]	10
4.1.3 Member Function Documentation	10
4.1.3.1 getEgz()	10
4.1.3.2 getGal()	10
4.1.3.3 getGalmed()	10
4.1.3.4 getGalvid()	10
4.1.3.5 getMed()	10
4.1.3.6 getPav()	10
4.1.3.7 getPaz()	11
4.1.3.8 getVar()	11
4.1.3.9 getVid()	11
4.1.3.10 operator=() [1/2]	11
4.1.3.11 operator=() [2/2]	11
4.1.3.12 setEgz()	11
4.1.3.13 setGal()	11
4.1.3.14 setGalmed()	12
4.1.3.15 setGalvid()	12
4.1.3.16 setMed()	12
4.1.3.17 setPav()	12
4.1.3.18 setPaz()	12
4.1.3.19 setVar()	12
4.1.3.20 setVid()	12
4.1.4 Friends And Related Symbol Documentation	13
4.1.4.1 operator<<	13
4.1.4.2 operator>>	13
4.2 Vektorius< T > Class Template Reference	13

4.2.1 Member Typedef Documentation	14
4.2.1.1 const_iterator	14
4.2.1.2 const_reverse_iterator	14
4.2.1.3 iterator	15
4.2.1.4 reverse_iterator	15
4.2.1.5 size_type	15
4.2.1.6 value_type	15
4.2.2 Constructor & Destructor Documentation	15
4.2.2.1 Vektorius() [1/5]	15
4.2.2.2 Vektorius() [2/5]	15
4.2.2.3 Vektorius() [3/5]	15
4.2.2.4 Vektorius() [4/5]	15
4.2.2.5 Vektorius() [5/5]	16
4.2.2.6 ~Vektorius()	16
4.2.3 Member Function Documentation	16
4.2.3.1 at() [1/2]	16
4.2.3.2 at() [2/2]	16
4.2.3.3 back() [1/2]	16
4.2.3.4 back() [2/2]	16
4.2.3.5 begin() [1/2]	16
4.2.3.6 begin() [2/2]	16
4.2.3.7 capacity()	17
4.2.3.8 cbegin()	17
4.2.3.9 cend()	17
4.2.3.10 clear()	17
4.2.3.11 crbegin()	17
4.2.3.12 crend()	17
4.2.3.13 data() [1/2]	17
4.2.3.14 data() [2/2]	17
4.2.3.15 empty()	17
4.2.3.16 end() [1/2]	18
4.2.3.17 end() [2/2]	18
4.2.3.18 erase() [1/2]	18
4.2.3.19 erase() [2/2]	18
4.2.3.20 front() [1/2]	18
4.2.3.21 front() [2/2]	18
4.2.3.22 insert()	18
4.2.3.23 max_size()	18
4.2.3.24 operator!=(())	19
4.2.3.25 operator<()	19
4.2.3.26 operator<=()	19
4.2.3.27 operator=() [1/2]	19

4.2.3.28 operator=() [2/2]	19
4.2.3.29 operator==()	19
4.2.3.30 operator>()	19
4.2.3.31 operator>=()	19
4.2.3.32 operator[]() [1/2]	20
4.2.3.33 operator[]() [2/2]	20
4.2.3.34 pop_back()	20
4.2.3.35 print()	20
4.2.3.36 push_back()	20
4.2.3.37 rbegin() [1/2]	20
4.2.3.38 rbegin() [2/2]	20
4.2.3.39 rend() [1/2]	20
4.2.3.40 rend() [2/2]	21
4.2.3.41 reserve()	21
4.2.3.42 resize()	21
4.2.3.43 shrink_to_fit()	21
4.2.3.44 size()	21
4.2.3.45 swap() [1/2]	21
4.2.3.46 swap() [2/2]	21
4.3 Zmogus Class Reference	22
4.3.1 Detailed Description	22
4.3.2 Constructor & Destructor Documentation	22
4.3.2.1 Zmogus() [1/2]	22
4.3.2.2 Zmogus() [2/2]	23
4.3.2.3 ~Zmogus()	23
4.3.3 Member Function Documentation	23
4.3.3.1 getPav()	23
4.3.3.2 getVar()	23
4.3.3.3 setPav()	23
4.3.3.4 setVar()	23
4.3.4 Member Data Documentation	23
4.3.4.1 pavarde_	23
4.3.4.2 vardas_	23
<b>5 File Documentation</b>	<b>25</b>
5.1 biblioteka.h File Reference	25
5.1.1 Detailed Description	25
5.2 biblioteka.h	25
5.3 funkcijos.cpp File Reference	26
5.3.1 Detailed Description	27
5.3.2 Function Documentation	27
5.3.2.1 arSveikasisSk()	27

5.3.2.2 arZodis()	27
5.3.2.3 FailuGeneravimas()	27
5.3.2.4 generavimasPav()	27
5.3.2.5 generavimasPaz()	27
5.3.2.6 generavimasVard()	28
5.3.2.7 Mediana()	28
5.3.2.8 rezrikiavimas()	28
5.3.2.9 rikiuotiMed()	28
5.3.2.10 rikiuotiPavarde()	28
5.3.2.11 rikiuotiVarda()	28
5.3.2.12 rikiuotiVid()	28
5.3.2.13 RusiavimasDviGrupes()	28
5.3.2.14 RusiavimasDviGrupes2()	29
5.3.2.15 RusiavimasDviGrupes3()	29
5.3.2.16 skaityti()	29
5.3.2.17 spausdinti()	29
5.3.2.18 spausdinti_skaitomus_duomenis()	29
5.3.2.19 spausdintiLaimingiVargsai()	29
5.3.2.20 testas()	29
5.3.2.21 Vidurkis()	30
5.4 funkcijos.h File Reference	30
5.4.1 Detailed Description	30
5.4.2 Function Documentation	31
5.4.2.1 arSveikasisSk()	31
5.4.2.2 arZodis()	31
5.4.2.3 FailuGeneravimas()	31
5.4.2.4 generavimasPav()	31
5.4.2.5 generavimasPaz()	31
5.4.2.6 generavimasVard()	31
5.4.2.7 Mediana()	31
5.4.2.8 rezrikiavimas()	31
5.4.2.9 rikiuotiMed()	32
5.4.2.10 rikiuotiPavarde()	32
5.4.2.11 rikiuotiVarda()	32
5.4.2.12 rikiuotiVid()	32
5.4.2.13 RusiavimasDviGrupes()	32
5.4.2.14 RusiavimasDviGrupes2()	32
5.4.2.15 RusiavimasDviGrupes3()	32
5.4.2.16 skaityti()	33
5.4.2.17 spausdinti()	33
5.4.2.18 spausdinti_skaitomus_duomenis()	33
5.4.2.19 spausdintiLaimingiVargsai()	33

---

5.4.2.20 testas()	33
5.4.2.21 Vidurkis()	33
5.5 funkcijos.h	34
5.6 v3.cpp File Reference	36
5.6.1 Detailed Description	36
5.6.2 Function Documentation	36
5.6.2.1 main()	36
5.7 vektorius.h File Reference	36
5.7.1 Detailed Description	37
5.8 vektorius.h	37
<b>Index</b>	<b>43</b>





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Vektorius< T > . . . . .	13
Vektorius< int > . . . . .	13
Zmogus . . . . .	22
Vartotojas . . . . .	7



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">Vartotojas</a>	Atstovauja vartotoja . . . . .	<a href="#">7</a>
<a href="#">Vektorius&lt; T &gt;</a>	. . . . .	<a href="#">13</a>
<a href="#">Zmogus</a>	Atstovauja zmogu su vardu ir pavarde . . . . .	<a href="#">22</a>



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">biblioteka.h</a>		
	Biblioteku deklaracija . . . . .	25
<a href="#">funkcijos.cpp</a>		
	Pagalbiniu funkciju failo vykdymas . . . . .	26
<a href="#">funkcijos.h</a>		
	Zmogus ir Vartotojas klases deklaracija ir funkciju reiksmiu priskyrimas . . . . .	30
<a href="#">v3.cpp</a>		
	Pagrindinio failo vykdymas . . . . .	36
<a href="#">vektorius.h</a>		
	Custom vektoriaus klase . . . . .	36



## Chapter 4

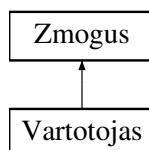
# Class Documentation

### 4.1 Vartotojas Class Reference

atstovauja vartotoja

```
#include <funkcijos.h>
```

Inheritance diagram for Vartotojas:



#### Public Member Functions

- [Vartotojas](#) ()  
*Vartotojas* klasės konstruktorius su nulinėmis reikšmėmis.
- [Vartotojas](#) (const string vardas, const string pavarde, const [Vektorius](#)< int > &nd, int egz, double vid, double gal, double med, double galmed, double galvid)  
*Vartotojas* klasės konstruktorius su parametrais.
- [~Vartotojas](#) ()  
*Vartotojas* klasės destruktorius.
- [Vartotojas](#) (const [Vartotojas](#) &other)  
*kopijavimo konstruktorius.*
- [Vartotojas](#) & [operator=](#) (const [Vartotojas](#) &other)  
*copy assignment operatorius.*
- [Vartotojas](#) ([Vartotojas](#) &&other) noexcept  
*move (perkėlimo) konstruktorius.*
- [Vartotojas](#) & [operator=](#) ([Vartotojas](#) &&other) noexcept  
*move assignment operatorius.*
- void [setPaz](#) (int paz)  
*Nustato namų darbų pažymius.*
- void [setVid](#) (double vidurkis)  
*Nustato namų darbų pažymių vidurkį.*

- void [setMed](#) (double med)  
*Nustato namų darbų pažymių medianą.*
- void [setEgz](#) (int egz)  
*Nustato egzamino rezultatą.*
- void [setGal](#) (double gal)  
*Nustato galutinį pažymį.*
- void [setGalvid](#) (double galv)  
*Nustato galutinį pažymį su namų darbų vidurkiu.*
- void [setGalmed](#) (double galm)  
*Nustato galutinį pažymį su namų darbų mediana.*
- void [setVar](#) (const std::string &vard)  
*Nustato studento vardą.*
- void [setPav](#) (const std::string &pav)  
*Nustato studento pavardę.*
- const [Vektorius](#)< int > & [getPaz](#) () const  
*Grąžina namų darbų pažymius.*
- int [getEgz](#) () const  
*Grąžina egzamino rezultatą.*
- double [getVid](#) () const  
*Grąžina namų darbų pažymių vidurkį.*
- double [getGal](#) () const  
*Grąžina galutinį pažymį.*
- double [getMed](#) () const  
*Grąžina namų darbų medianą.*
- double [getGalmed](#) () const  
*Grąžina galutinį pažymį su mediana.*
- double [getGalvid](#) () const  
*Grąžina galutinį pažymį su vidurkiu.*
- string [getVar](#) () const override  
*Grąžina vardą.*
- string [getPav](#) () const override  
*Grąžina pavardę.*

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

- [Zmogus](#) ()=default  
*default [Zmogus](#) klasės konstruktorius.*
- [Zmogus](#) (string vardas, string pavarde)  
*[Zmogus](#) klasės konstruktorius su parametrais.*
- virtual [~Zmogus](#) ()
- virtual void [setVar](#) (string vard)
- virtual void [setPav](#) (string pav)

## Friends

- ostream & [operator<<](#) (ostream &out, const [Vartotojas](#) &vart)  
*Išvesties perdengimo operatoriai.*
- istream & [operator>>](#) (istream &in, [Vartotojas](#) &vart)  
*Išvesties perdengimo operatoriai.*



## Additional Inherited Members

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

- string [vardas\\_](#)  
*Studento vardas.*
- string [pavarde\\_](#)  
*Studento pavardė.*

#### 4.1.1 Detailed Description

atstovauja vartotoja

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 Vartotojas() [1/4]

```
Vartotojas::Vartotojas ( ) [inline]
```

[Vartotojas](#) klasės konstruktorius su nulinėmis reikšmėmis.

##### 4.1.2.2 Vartotojas() [2/4]

```
Vartotojas::Vartotojas (
    const string vardas,
    const string pavarde,
    const Vektorius< int > & nd,
    int egz,
    double vid,
    double gal,
    double med,
    double galmed,
    double galvid ) [inline]
```

[Vartotojas](#) klasės konstruktorius su parametrais.

##### 4.1.2.3 ~Vartotojas()

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

[Vartotojas](#) klasės destruktorius.

##### 4.1.2.4 Vartotojas() [3/4]

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

kopijavimo konstruktorius.

#### 4.1.2.5 Vartotojas() [4/4]

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

move (perkėlimo) konstruktorius.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 getEgz()

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

Grąžina egzamino rezultatą.

#### 4.1.3.2 getGal()

```
double Vartotojas::getGal ( ) const [inline]
```

Grąžina galutinį pažymį.

#### 4.1.3.3 getGalmed()

```
double Vartotojas::getGalmed ( ) const [inline]
```

Grąžina galutinį pažymį su mediana.

#### 4.1.3.4 getGalvid()

```
double Vartotojas::getGalvid ( ) const [inline]
```

Grąžina galutinį pažymį su vidurkiu.

#### 4.1.3.5 getMed()

```
double Vartotojas::getMed ( ) const [inline]
```

Grąžina namų darbų medianą.

#### 4.1.3.6 getPav()

```
string Vartotojas::getPav ( ) const [inline], [override], [virtual]
```

Grąžina pavardę.

Implements [Zmogus](#).

#### 4.1.3.7 getPaz()

```
const Vektorius< int > & Vartotojas::getPaz ( ) const [inline]
```

Grąžina namų darbų pažymius.

#### 4.1.3.8 getVar()

```
string Vartotojas::getVar ( ) const [inline], [override], [virtual]
```

Grąžina vardą.

Implements [Zmogus](#).

#### 4.1.3.9 getVid()

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

Grąžina namų darbų pažymių vidurkį.

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

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

copy assignment operatorius.

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

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

move assignment operatorius.

#### 4.1.3.12 setEgz()

```
void Vartotojas::setEgz (
    int egz ) [inline]
```

Nustato egzamino rezultatą.

#### 4.1.3.13 setGal()

```
void Vartotojas::setGal (
    double gal ) [inline]
```

Nustato galutinį pažymį.

**4.1.3.14 setGalmed()**

```
void Vartotojas::setGalmed (
    double galmed ) [inline]
```

Nustato galutinį pažymį su namų darbų mediana.

**4.1.3.15 setGalvid()**

```
void Vartotojas::setGalvid (
    double galv ) [inline]
```

Nustato galutinį pažymį su namų darbų vidurkiu.

**4.1.3.16 setMed()**

```
void Vartotojas::setMed (
    double med ) [inline]
```

Nustato namų darbų pažymių medianą.

**4.1.3.17 setPav()**

```
void Vartotojas::setPav (
    const std::string & pav ) [inline]
```

Nustato studento vardą.

**4.1.3.18 setPaz()**

```
void Vartotojas::setPaz (
    int paz ) [inline]
```

Nustato namų darbų pažymius.

**4.1.3.19 setVar()**

```
void Vartotojas::setVar (
    const std::string & vard ) [inline]
```

Nustato studento vardą.

**4.1.3.20 setVid()**

```
void Vartotojas::setVid (
    double vidurkis ) [inline]
```

Nustato namų darbų pažymių vidurkį.

### 4.1.4 Friends And Related Symbol Documentation

#### 4.1.4.1 operator<<

```
ostream & operator<< (
    ostream & out,
    const Vartotojas & vart ) [friend]
```

Išvesties perdengimo operatoriai.

#### 4.1.4.2 operator>>

```
istream & operator>> (
    istream & in,
    Vartotojas & vart ) [friend]
```

Išvesties perdengimo operatoriai.

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

- [funkcijos.h](#)

## 4.2 Vektorius< T > Class Template Reference

```
#include <vektorius.h>
```

### Public Types

- [typedef T value\\_type](#)
- [typedef T \\* iterator](#)
- [typedef const T \\* const\\_iterator](#)
- [typedef size\\_t size\\_type](#)
- [typedef std::reverse\\_iterator< iterator > reverse\\_iterator](#)
- [typedef std::reverse\\_iterator< const\\_iterator > const\\_reverse\\_iterator](#)

### Public Member Functions

- [Vektorius \(\)](#)
- [Vektorius \(size\\_type dydis, const T &value=T\(\)\)](#)
- [Vektorius \(const Vektorius &kitas\)](#)
- [Vektorius \(Vektorius &&kitas\) noexcept](#)
- [Vektorius \(std::initializer\\_list< T > sarasas\)](#)
- [~Vektorius \(\)](#)
- [Vektorius & operator= \(const Vektorius &kitas\)](#)
- [Vektorius & operator= \(Vektorius &&kitas\)](#)
- [T & operator\[\] \(size\\_type indeksas\)](#)
- [const T & operator\[\] \(size\\_type indeksas\) const](#)
- [T & at \(size\\_type indeksas\)](#)
- [const T & at \(size\\_type indeksas\) const](#)

- `T & front ()`
- `const T & front () const`
- `T & back ()`
- `const T & back () const`
- `T * data () noexcept`
- `const T * data () const noexcept`
- `size_type size () const noexcept`
- `size_type capacity () const noexcept`
- `bool empty () const noexcept`
- `size_type max_size () const noexcept`
- `void reserve (size_type naujaTalpa)`
- `void shrink_to_fit ()`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `void clear () noexcept`
- `iterator insert (const_iterator pos, const T &value)`
- `iterator erase (const_iterator pos)`
- `iterator erase (const_iterator first, const_iterator last)`
- `void push_back (const T &x)`
- `void pop_back ()`
- `void resize (size_type new_size)`
- `void swap (Vektorius &other) noexcept`
- `bool operator== (const Vektorius< T > &other) const`
- `bool operator!= (const Vektorius< T > &other) const`
- `bool operator< (const Vektorius< T > &other) const`
- `bool operator<= (const Vektorius< T > &other) const`
- `bool operator> (const Vektorius< T > &other) const`
- `bool operator>= (const Vektorius< T > &other) const`
- `void swap (Vektorius< T > &x, Vektorius< T > &y)`
- `void print () const`

## 4.2.1 Member Typedef Documentation

### 4.2.1.1 const\_iterator

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

### 4.2.1.2 const\_reverse\_iterator

```
template<typename T >
typedef std::reverse_iterator<const_iterator> Vektorius< T >::const_reverse_iterator
```

#### 4.2.1.3 iterator

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

#### 4.2.1.4 reverse\_iterator

```
template<typename T >
typedef std::reverse_iterator<iterator> Vektorius< T >::reverse_iterator
```

#### 4.2.1.5 size\_type

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

#### 4.2.1.6 value\_type

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

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Vektorius() [1/5]

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

#### 4.2.2.2 Vektorius() [2/5]

```
template<typename T >
Vektorius< T >::Vektorius (
    size_type dydis,
    const T & value = T() ) [inline]
```

#### 4.2.2.3 Vektorius() [3/5]

```
template<typename T >
Vektorius< T >::Vektorius (
    const Vektorius< T > & kitas ) [inline]
```

#### 4.2.2.4 Vektorius() [4/5]

```
template<typename T >
Vektorius< T >::Vektorius (
    Vektorius< T > && kitas ) [inline], [noexcept]
```

#### 4.2.2.5 Vektorius() [5/5]

```
template<typename T >
Vektorius< T >::Vektorius (
    std::initializer_list< T > sarasas ) [inline]
```

#### 4.2.2.6 ~Vektorius()

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

### 4.2.3 Member Function Documentation

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

```
template<typename T >
T & Vektorius< T >::at (
    size_type indeksas ) [inline]
```

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

```
template<typename T >
const T & Vektorius< T >::at (
    size_type indeksas ) const [inline]
```

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

```
template<typename T >
T & Vektorius< T >::back ( ) [inline]
```

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

```
template<typename T >
const T & Vektorius< T >::back ( ) const [inline]
```

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

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

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

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



#### 4.2.3.7 capacity()

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

#### 4.2.3.8 cbegin()

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

#### 4.2.3.9 cend()

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

#### 4.2.3.10 clear()

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

#### 4.2.3.11 crbegin()

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

#### 4.2.3.12 crend()

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

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

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

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

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

#### 4.2.3.15 empty()

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

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

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

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

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

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

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

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

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

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

```
template<typename T >
T & Vektorius< T >::front ( ) [inline]
```

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

```
template<typename T >
const T & Vektorius< T >::front ( ) const [inline]
```

**4.2.3.22 insert()**

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

**4.2.3.23 max\_size()**

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

**4.2.3.24 operator"!="()**

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

**4.2.3.25 operator<()**

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

**4.2.3.26 operator<=()**

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

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

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

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

```
template<typename T >
Vektorius & Vektorius< T >::operator= (
    Vektorius< T > && kitas ) [inline]
```

**4.2.3.29 operator==(())**

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

**4.2.3.30 operator>()**

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

**4.2.3.31 operator>=()**

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

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

```
template<typename T >
T & Vektorius< T >::operator[] (
    size_type indeksas ) [inline]
```

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

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

**4.2.3.34 pop\_back()**

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

**4.2.3.35 print()**

```
template<typename T >
void Vektorius< T >::print ( ) const [inline]
```

**4.2.3.36 push\_back()**

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

**4.2.3.37 rbegin()** [1/2]

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

**4.2.3.38 rbegin()** [2/2]

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

**4.2.3.39 rend()** [1/2]

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

**4.2.3.40** `rend()` [2/2]

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

**4.2.3.41** `reserve()`

```
template<typename T >
void Vektorius< T >::reserve (
    size_type naujaTalpa ) [inline]
```

**4.2.3.42** `resize()`

```
template<typename T >
void Vektorius< T >::resize (
    size_type new_size ) [inline]
```

**4.2.3.43** `shrink_to_fit()`

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

**4.2.3.44** `size()`

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

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

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

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

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

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

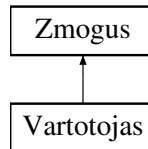
- [vektorius.h](#)

## 4.3 Zmogus Class Reference

atstovauja zmogu su vardu ir pavarde

```
#include <funkcijos.h>
```

Inheritance diagram for Zmogus:



### Public Member Functions

- [Zmogus](#) ()=default  
*default [Zmogus](#) klasės konstruktorius.*
- [Zmogus](#) (string vardas, string pavarde)  
*[Zmogus](#) klasės konstruktorius su parametrais.*
- virtual [~Zmogus](#) ()
- virtual void [setVar](#) (string vard)
- virtual void [setPav](#) (string pav)
- virtual string [getVar](#) () const =0
- virtual string [getPav](#) () const =0

### Protected Attributes

- string [vardas\\_](#)  
*Studento vardas.*
- string [pavarde\\_](#)  
*Studento pavardė.*

### 4.3.1 Detailed Description

atstovauja zmogu su vardu ir pavarde

### 4.3.2 Constructor & Destructor Documentation

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

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

default [Zmogus](#) klasės konstruktorius.

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

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

[Zmogus](#) klasės konstruktorius su parametrais.

#### 4.3.2.3 ~Zmogus()

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

### 4.3.3 Member Function Documentation

#### 4.3.3.1 getPav()

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

Implemented in [Vartotojas](#).

#### 4.3.3.2 getVar()

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

Implemented in [Vartotojas](#).

#### 4.3.3.3 setPav()

```
virtual void Zmogus::setPav (
    string pav ) [inline], [virtual]
```

#### 4.3.3.4 setVar()

```
virtual void Zmogus::setVar (
    string vard ) [inline], [virtual]
```

### 4.3.4 Member Data Documentation

#### 4.3.4.1 pavarde\_

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

Studento pavardė.

#### 4.3.4.2 vardas\_

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

Studento vardas.

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

- [funkcijos.h](#)





## Chapter 5

# File Documentation

### 5.1 biblioteka.h File Reference

biblioteku deklaracija

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <fstream>
#include <sstream>
#include <chrono>
#include <cctype>
#include <random>
#include <list>
#include <deque>
#include <utility>
#include <assert.h>
```

#### 5.1.1 Detailed Description

biblioteku deklaracija

### 5.2 biblioteka.h

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

```
00001 //
00002 //  biblioteka.h
00003 //  V3.0
00004 //
00005 //  Created by Kamilè Zobèlaitè on 2024-05-15.
00006 //
00007
00008 //
00009 //  biblioteka.h
00010 //  v2.0
```

```

00011 //
00012 // Created by Kamilė Zobelaitė on 2024-04-29.
00013 //
00019 #ifndef biblioteka_h
00020 #define biblioteka_h
00021
00022 #include <iostream>
00023 #include <iomanip>
00024 #include <algorithm>
00025 #include <cstdlib>
00026 #include <ctime>
00027 #include <vector>
00028 #include <fstream>
00029 #include <sstream>
00030 #include <chrono>
00031 #include <cctype>
00032 #include <random>
00033 #include <list>
00034 #include <deque>
00035 #include <utility>
00036 #include <assert.h>
00037
00038
00039 using std::cout;
00040 using std::cin;
00041 using std::endl;
00042 using std::left;
00043 using std::srand;
00044 using std::rand;
00045 using std::vector;
00046 using std::setw;
00047 using std::fixed;
00048 using std::setprecision;
00049 using std::string;
00050 using std::sort;
00051 using std::stringstream;
00052 using std::ifstream;
00053 using std::ofstream;
00054 using std::exception;
00055 using std::numeric_limits;
00056 using std::streamsize;
00057 using std::runtime_error;
00058 using std::cerr;
00059 using std::list;
00060 using std::deque;
00061 using std::move;
00062 using std::ostream;
00063 using std::istream;
00064 using std::accumulate;
00065 #endif /* biblioteka_h */

```

## 5.3 funkcijos.cpp File Reference

Pagalbinių funkcijų failo vykdymas.

```
#include "funkcijos.h"
```

### Functions

- double [Vidurkis](#) (double suma, int nariai)
- double [Mediana](#) ([Vektorius](#)< int > paz, int nariai)
- void [spausdinti](#) (int rnkts, [Vektorius](#)< [Vartotojas](#) > &vart, int n)
- double [generavimasPaz](#) ()
- string [generavimasVard](#) (int pas)
- string [generavimasPav](#) (int pas)
- void [skaityti](#) ([Vektorius](#)< [Vartotojas](#) > &vart, string pavadinimas, int vm)
- void [rezikiavimas](#) ([Vektorius](#)< [Vartotojas](#) > &vart)
- void [spausdinti\\_skaitomus\\_duomenis](#) ([Vektorius](#)< [Vartotojas](#) > &vart)
- bool [rikiuotiVarda](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)

- bool [rikiuotiPavarde](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [rikiuotiVid](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [rikiuotiMed](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [arZodis](#) (string *tekstas*)
- bool [arSveikasisSk](#) (string *tekstas*)
- void [FailuGeneravimas](#) (int *studSk*)
- void [RusiavimasDviGrupes](#) ([Vektorius](#)< [Vartotojas](#) > &vart, [Vektorius](#)< [Vartotojas](#) > &vargsai, [Vektorius](#)< [Vartotojas](#) > &laimingi, int *vm*)
- void [spausdintiLaimingiVargsai](#) ([Vektorius](#)< [Vartotojas](#) > &vargsai, [Vektorius](#)< [Vartotojas](#) > &laimingi, int *vm*)
- void [RusiavimasDviGrupes2](#) ([Vektorius](#)< [Vartotojas](#) > &vart, [Vektorius](#)< [Vartotojas](#) > &vargsai, int *vm*)
- void [RusiavimasDviGrupes3](#) ([Vektorius](#)< [Vartotojas](#) > &vart, [Vektorius](#)< [Vartotojas](#) > &vargsai, int *vm*)
- void [testas](#) ()

### 5.3.1 Detailed Description

Pagalbiniu funkciju failo vykdymas.

### 5.3.2 Function Documentation

#### 5.3.2.1 [arSveikasisSk\(\)](#)

```
bool arSveikasisSk (
    string tekstas )
```

#### 5.3.2.2 [arZodis\(\)](#)

```
bool arZodis (
    string tekstas )
```

#### 5.3.2.3 [FailuGeneravimas\(\)](#)

```
void FailuGeneravimas (
    int studSk )
```

#### 5.3.2.4 [generavimasPav\(\)](#)

```
string generavimasPav (
    int pas )
```

#### 5.3.2.5 [generavimasPaz\(\)](#)

```
double generavimasPaz ( )
```

### 5.3.2.6 generavimasVard()

```
string generavimasVard (
    int pas )
```

### 5.3.2.7 Mediana()

```
double Mediana (
    Vektorius< int > paz,
    int nariai )
```

### 5.3.2.8 rezrkiavimas()

```
void rezrkiavimas (
    Vektorius< Vartotojas > & vart )
```

### 5.3.2.9 rikiuotiMed()

```
bool rikiuotiMed (
    const Vartotojas & a,
    const Vartotojas & b )
```

### 5.3.2.10 rikiuotiPavarde()

```
bool rikiuotiPavarde (
    const Vartotojas & a,
    const Vartotojas & b )
```

### 5.3.2.11 rikiuotiVarda()

```
bool rikiuotiVarda (
    const Vartotojas & a,
    const Vartotojas & b )
```

### 5.3.2.12 rikiuotiVid()

```
bool rikiuotiVid (
    const Vartotojas & a,
    const Vartotojas & b )
```

### 5.3.2.13 RusiavimasDviGrupes()

```
void RusiavimasDviGrupes (
    Vektorius< Vartotojas > & vart,
    Vektorius< Vartotojas > & vargsai,
    Vektorius< Vartotojas > & laimingi,
    int vm )
```

#### 5.3.2.14 RusiavimasDviGrupes2()

```
void RusiavimasDviGrupes2 (
    Vektorius< Vartotojas > & vart,
    Vektorius< Vartotojas > & vargsai,
    int vm )
```

#### 5.3.2.15 RusiavimasDviGrupes3()

```
void RusiavimasDviGrupes3 (
    Vektorius< Vartotojas > & vart,
    Vektorius< Vartotojas > & vargsai,
    int vm )
```

#### 5.3.2.16 skaityti()

```
void skaityti (
    Vektorius< Vartotojas > & vart,
    string pavadinimas,
    int vm )
```

#### 5.3.2.17 spausdinti()

```
void spausdinti (
    int rnkts,
    Vektorius< Vartotojas > & vart,
    int n )
```

#### 5.3.2.18 spausdinti\_skaitomus\_duomenis()

```
void spausdinti_skaitomus_duomenis (
    Vektorius< Vartotojas > & vart )
```

#### 5.3.2.19 spausdintiLaimingiVargsai()

```
void spausdintiLaimingiVargsai (
    Vektorius< Vartotojas > & vargsai,
    Vektorius< Vartotojas > & laimingi,
    int vm )
```

#### 5.3.2.20 testas()

```
void testas ( )
```

### 5.3.2.21 Vidurkis()

```
double Vidurkis (  
    double suma,  
    int nariai )
```

## 5.4 funkcijos.h File Reference

[Zmogus](#) ir [Vartotojas](#) klases deklaracija ir funkciju reiksmiu priskyrimas.

```
#include "biblioteka.h"  
#include "vektorius.h"
```

### Classes

- class [Zmogus](#)  
*atstovauja zmogu su vardu ir pavarde*
- class [Vartotojas](#)  
*atstovauja vartotoja*

### Functions

- double [Vidurkis](#) (double suma, int nariai)
- double [Mediana](#) ([Vektorius](#)< int > paz, int nariai)
- void [spausdinti](#) (int rnkts, [Vektorius](#)< [Vartotojas](#) > &vart, int n)
- double [generavimasPaz](#) ()
- string [generavimasVard](#) (int pas)
- string [generavimasPav](#) (int pas)
- void [skaityti](#) ([Vektorius](#)< [Vartotojas](#) > &vart, string pavadinimas, int vm)
- void [rezikiavimas](#) ([Vektorius](#)< [Vartotojas](#) > &vart)
- void [spausdinti\\_skaitomus\\_duomenis](#) ([Vektorius](#)< [Vartotojas](#) > &vart)
- bool [rikiuotiVarda](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [rikiuotiPavarde](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [rikiuotiVid](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [rikiuotiMed](#) (const [Vartotojas](#) &a, const [Vartotojas](#) &b)
- bool [arZodis](#) (string tekstas)
- bool [arSveikasisSk](#) (string tekstas)
- void [FailuGeneravimas](#) (int studSk)
- void [RusiavimasDviGrupes](#) ([Vektorius](#)< [Vartotojas](#) > &vart, [Vektorius](#)< [Vartotojas](#) > &vargsai, [Vektorius](#)< [Vartotojas](#) > &laimingi, int vm)
- void [spausdintiLaimingiVargsai](#) ([Vektorius](#)< [Vartotojas](#) > &vargsai, [Vektorius](#)< [Vartotojas](#) > &laimingi, int vm)
- void [RusiavimasDviGrupes2](#) ([Vektorius](#)< [Vartotojas](#) > &vart, [Vektorius](#)< [Vartotojas](#) > &vargsai, int vm)
- void [RusiavimasDviGrupes3](#) ([Vektorius](#)< [Vartotojas](#) > &vart, [Vektorius](#)< [Vartotojas](#) > &vargsai, int vm)
- void [testas](#) ()

### 5.4.1 Detailed Description

[Zmogus](#) ir [Vartotojas](#) klases deklaracija ir funkciju reiksmiu priskyrimas.

## 5.4.2 Function Documentation

### 5.4.2.1 arSveikasisSk()

```
bool arSveikasisSk (
    string tekstas )
```

### 5.4.2.2 arZodis()

```
bool arZodis (
    string tekstas )
```

### 5.4.2.3 FailuGeneravimas()

```
void FailuGeneravimas (
    int studSk )
```

### 5.4.2.4 generavimasPav()

```
string generavimasPav (
    int pas )
```

### 5.4.2.5 generavimasPaz()

```
double generavimasPaz ( )
```

### 5.4.2.6 generavimasVard()

```
string generavimasVard (
    int pas )
```

### 5.4.2.7 Mediana()

```
double Mediana (
    Vektorius< int > paz,
    int nariai )
```

### 5.4.2.8 rezrkiavimas()

```
void rezrkiavimas (
    Vektorius< Vartotojas > & vart )
```

#### 5.4.2.9 rikiuotiMed()

```
bool rikiuotiMed (
    const Vartotojas & a,
    const Vartotojas & b )
```

#### 5.4.2.10 rikiuotiPavarde()

```
bool rikiuotiPavarde (
    const Vartotojas & a,
    const Vartotojas & b )
```

#### 5.4.2.11 rikiuotiVarda()

```
bool rikiuotiVarda (
    const Vartotojas & a,
    const Vartotojas & b )
```

#### 5.4.2.12 rikiuotiVid()

```
bool rikiuotiVid (
    const Vartotojas & a,
    const Vartotojas & b )
```

#### 5.4.2.13 RusiavimasDviGrupes()

```
void RusiavimasDviGrupes (
    Vektorius< Vartotojas > & vart,
    Vektorius< Vartotojas > & vargsai,
    Vektorius< Vartotojas > & laimingi,
    int vm )
```

#### 5.4.2.14 RusiavimasDviGrupes2()

```
void RusiavimasDviGrupes2 (
    Vektorius< Vartotojas > & vart,
    Vektorius< Vartotojas > & vargsai,
    int vm )
```

#### 5.4.2.15 RusiavimasDviGrupes3()

```
void RusiavimasDviGrupes3 (
    Vektorius< Vartotojas > & vart,
    Vektorius< Vartotojas > & vargsai,
    int vm )
```



#### 5.4.2.16 skaityti()

```
void skaityti (
    Vektorius< Vartotojas > & vart,
    string pavadinimas,
    int vm )
```

#### 5.4.2.17 spausdinti()

```
void spausdinti (
    int rnkts,
    Vektorius< Vartotojas > & vart,
    int n )
```

#### 5.4.2.18 spausdinti\_skaitomus\_duomenis()

```
void spausdinti_skaitomus_duomenis (
    Vektorius< Vartotojas > & vart )
```

#### 5.4.2.19 spausdintiLaimingiVargsai()

```
void spausdintiLaimingiVargsai (
    Vektorius< Vartotojas > & vargsai,
    Vektorius< Vartotojas > & laimingi,
    int vm )
```

#### 5.4.2.20 testas()

```
void testas ( )
```

#### 5.4.2.21 Vidurkis()

```
double Vidurkis (
    double suma,
    int nariai )
```

## 5.5 funkcijos.h

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

```

00001 //
00002 //  funkcijos.h
00003 //  V3.0
00004 //
00005 //  Created by Kamilė Zobėlaitė on 2024-05-15.
00006 //
00007
00008 //
00009 //  funkcijos.h
00010 //  v2.0
00011 //
00012 //  Created by Kamilė Zobėlaitė on 2024-04-29.
00013 //
00014
00019 #ifndef funkcijos_h
00020 #define funkcijos_h
00021
00022 #include "biblioteka.h"
00023 #include "vektorius.h"
00028 class Zmogus
00029 {
00030 protected:
00031
00032     string vardas_;
00033     string pavarde_;
00034 public:
00038     Zmogus() = default;
00042     Zmogus(string vardas, string pavarde)
00043     : vardas_(vardas), pavarde_(pavarde) {}
00044     virtual ~Zmogus() {}
00045     // virtual void kazkas() const = 0;
00046     virtual void setVar (string vard) {
00047         vardas_=vard;
00048     }
00049     virtual void setPav (string pav) {
00050         pavarde_=pav;
00051     }
00052
00053     virtual string getVar() const = 0;
00054     virtual string getPav() const = 0;
00055 };
00060 class Vartotojas : public Zmogus
00061 {
00062 private:
00063     Vektorius<int> nd_;
00064     int egz_;
00065     double vid_;
00066     double gal_;
00067     double med_;
00068     double galmed_;
00069     double galvid_;
00070 public:
00074     Vartotojas() : vid_(0.0), gal_(0.0), med_(0.0), galmed_(0.0), galvid_(0.0) {}
00078     Vartotojas(const string vardas, const string pavarde, const Vektorius<int>& nd, int egz, double
vid, double gal, double med, double galmed, double galvid)
00079     : Zmogus(vardas, pavarde), nd_(nd), egz_(egz), vid_(vid), gal_(gal), med_(med), galmed_(galmed),
galvid_(galvid) {}
00080     // destruktorius
00084     ~Vartotojas() {
00085         //cout << "Objektas sunaikintas" << endl;
00086         nd_.clear();
00087     }
00088     // copy konstruktorius
00091     Vartotojas(const Vartotojas& other)
00092     : Zmogus(other.getVar(), other.getPav()), nd_(other.nd_), egz_(other.egz_), vid_(other.vid_),
gal_(other.gal_), med_(other.med_), galmed_(other.galmed_), galvid_(other.galvid_) {}
00093     //void kazkas () const override{}
00094
00095     // Copy assignment operatorius
00098     Vartotojas& operator=(const Vartotojas& other) {
00099         if (this != &other) {
00100             Zmogus::setVar(other.getVar());
00101             Zmogus::setPav(other.getPav());
00102             nd_ = other.nd_;
00103             egz_ = other.egz_;
00104             vid_ = other.vid_;
00105             gal_ = other.gal_;
00106             med_ = other.med_;
00107             galmed_ = other.galmed_;
00108             galvid_ = other.galvid_;
00109         }
00110         return *this;

```

```

00111     }
00112     // move konstruktorius
00113     Vartotojas(Vartotojas&& other) noexcept
00114     : Zmogus(std::move(other.vardas_), std::move(other.pavarde_), nd_(std::move(other.nd_)),
    egz_(other.egz_), vid_(other.vid_), gal_(other.gal_), med_(other.med_), galmed_(other.galmed_),
    galvid_(other.galvid_) {
00117         other.egz_ = 0;
00118         other.vid_ = 0.0;
00119         other.gal_ = 0.0;
00120         other.med_ = 0.0;
00121         other.galmed_ = 0.0;
00122         other.galvid_ = 0.0;
00123     }
00124     // move assignment operatorius
00125     Vartotojas& operator=(Vartotojas&& other) noexcept {
00126         if (this != &other) {
00127             Zmogus::setVar(std::move(other.vardas_));
00128             Zmogus::setPav(std::move(other.pavarde_));
00129             nd_ = std::move(other.nd_);
00130             egz_ = other.egz_;
00131             vid_ = other.vid_;
00132             gal_ = other.gal_;
00133             med_ = other.med_;
00134             galmed_ = other.galmed_;
00135             galvid_ = other.galvid_;
00136             other.egz_ = 0;
00137             other.vid_ = 0.0;
00138             other.gal_ = 0.0;
00139             other.med_ = 0.0;
00140             other.galmed_ = 0.0;
00141             other.galvid_ = 0.0;
00142         }
00143         return *this;
00144     }
00145     friend ostream& operator<<(ostream& out, const Vartotojas &vart){
00146         out << left << setw(20) << vart.vardas_ << setw(20) << vart.pavarde_ << setw(20) << fixed <<
    setprecision(2) << vart.gal_ << endl;
00147         return out;
00148     }
00149     friend istream& operator>>(istream& in, Vartotojas &vart){
00150         in >> vart.vardas_ >> vart.pavarde_;
00151         int paz;
00152         Vektorius<int> pzm;
00153         while(in >> paz)
00154         {
00155             vart.nd_.push_back(paz);
00156         }
00157         if(!vart.nd_.empty())
00158         {
00159             vart.egz_ = vart.nd_.back();
00160             vart.nd_.pop_back();
00161         }
00162         return in;
00163     }
00164     void setPaz(int paz){
00165         nd_.push_back(paz);
00166     }
00167     void setVid(double vidurkis){
00168         vid_=vidurkis;
00169     }
00170     void setMed(double med){
00171         med_=med;
00172     }
00173     void setEgz (int egz){
00174         egz_ = egz;
00175     }
00176     void setGal(double gal){
00177         gal_ = gal;
00178     }
00179     void setGalvid(double galv){
00180         galvid_=galv;
00181     }
00182     void setGalmed(double galm){
00183         galmed_=galm;
00184     }
00185     void setVar(const std::string& vard) { Zmogus::setVar(vard); }
00186     void setPav(const std::string& pav) { Zmogus::setPav(pav); }
00187     const Vektorius<int>& getPaz() const { return nd_; }
00188     int getEgz() const { return egz_; }
00189     double getVid() const { return vid_; }
00190     double getGal() const { return gal_; }
00191     double getMed() const { return med_; }
00192     double getGalmed() const { return galmed_; }
00193     double getGalvid() const { return galvid_; }
00194     string getVar() const override { return vardas_; }

```

```

00240     string getPav() const override { return pavarde_; }
00241
00242
00243 };
00244
00245 double Vidurkis(double suma, int nariai);
00246 double Mediana(Vektorius<int> paz, int nariai);
00247 void spausdinti(int rnkts, Vektorius<Vartotojas>& vart, int n);
00248 double generavimasPaz();
00249 string generavimasVard(int pas);
00250 string generavimasPav(int pas);
00251 void skaityti(Vektorius<Vartotojas>& vart, string pavadinimas, int vm);
00252 void rezikiavimas(Vektorius<Vartotojas>& vart);
00253 void spausdinti_skaitomus_duomenis(Vektorius<Vartotojas>& vart);
00254 bool rikiuotiVarda(const Vartotojas &a, const Vartotojas &b);
00255 bool rikiuotiPavarde(const Vartotojas &a, const Vartotojas &b);
00256 bool rikiuotiVid(const Vartotojas &a, const Vartotojas &b);
00257 bool rikiuotiMed(const Vartotojas &a, const Vartotojas &b);
00258 bool arZodis(string tekstas);
00259 bool arSveikasisSk(string tekstas);
00260 void FailuGeneravimas (int studSk);
00261 void RusiavimasDviGrupes(Vektorius<Vartotojas>& vart, Vektorius<Vartotojas>& vargsai,
    Vektorius<Vartotojas>& laimingi, int vm);
00262 void spausdintiLaimingiVargsai (Vektorius<Vartotojas>& vargsai, Vektorius<Vartotojas>& laimingi, int
    vm);
00263 void RusiavimasDviGrupes2(Vektorius<Vartotojas>& vart, Vektorius<Vartotojas>& vargsai, int vm);
00264 void RusiavimasDviGrupes3(Vektorius<Vartotojas>& vart, Vektorius<Vartotojas>& vargsai, int vm);
00265 void testas();
00266
00267
00268 #endif /* funkcijos_h */

```

## 5.6 v3.cpp File Reference

Pagrindinio failo vykdymas.

```

#include "funkcijos.h"
#include "vektorius.h"

```

### Functions

- int [main](#) ()

### 5.6.1 Detailed Description

Pagrindinio failo vykdymas.

### 5.6.2 Function Documentation

#### 5.6.2.1 main()

```
int main ( )
```

## 5.7 vektorius.h File Reference

custom vektoriaus klase

```
#include <iterator>
```

## Classes

- class [Vektorius< T >](#)

## 5.7.1 Detailed Description

custom vektoriaus klase

## 5.8 vektorius.h

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

```

00001 //
00002 //   vector.h
00003 //   V3.0
00004 //
00005 //   Created by Kamilė Zobėlaitė on 2024-05-15.
00006 //
00007
00008 #ifndef vektorius_h
00009 #define vektorius_h
00010 #include <iterator>
00011 template <typename T>
00012 class Vektorius
00013 {
00014 private:
00015     T* mduom_; // Dinaminis masyvas, saugantis duomenis
00016     size_t mdydis_; // Konteinerio dydis (dabar kiek elementu yra vektoriuje)
00017     size_t mtalpa_; // Talpa (kiek elementų gali būti saugoma)
00018 public:
00019     // member types
00020     typedef T value_type;
00021     typedef T* iterator;
00022     typedef const T* const_iterator;
00023     typedef size_t size_type;
00024     typedef std::reverse_iterator<iterator> reverse_iterator;
00025     typedef std::reverse_iterator<const_iterator> const_reverse_iterator;
00026
00027     // MEMBER FUNCTIONS
00028     // Konstruktoriai
00029     // (1) Empty container constructor (default constructor)
00030     Vektorius() : mduom_(nullptr), mdydis_(0), mtalpa_(0) {}
00031
00032     // (2) Fill constructor
00033     Vektorius(size_type dydis, const T& value = T())
00034         : mduom_(new T[dydis]), mdydis_(dydis), mtalpa_(dydis)
00035     {
00036         std::fill(mduom_, mduom_ + mdydis_, value);
00037     }
00038
00039     // (3) Range constructor
00040     template <typename InputIterator>
00041     Vektorius(InputIterator first, InputIterator last)
00042     {
00043         mdydis_ = std::distance(first, last);
00044         mtalpa_ = mdydis_;
00045         mduom_ = new T[mdydis_];
00046         std::copy(first, last, mduom_);
00047     }
00048
00049     // (3) Range constructor
00050     template <typename InputIterator>
00051     Vektorius(InputIterator first, InputIterator last)
00052         : mduom_(nullptr), mdydis_(0), mtalpa_(0)
00053     {
00054         while (first != last) {
00055             push_back(*first);
00056             first = std::next(first);
00057         }
00058     }
00059
00060     // (4) Copy constructor
00061     Vektorius(const Vektorius& kitas)
00062         : mduom_(new T[kitas.mdydis_]), mdydis_(kitas.mdydis_), mtalpa_(kitas.mtalpa_)
00063     {
00064         for(int i = 0; i != mdydis_; ++i)
00065         {

```

```

00069         mduom_[i]=kitas.mduom_[i];
00070     }
00071 }
00072
00073 // (5) Move constructor
00074 Vektorius(Vektorius&& kitas) noexcept
00075     : mduom_(kitas.mduom_), mdydis_(kitas.mdydis_), mtalpa_(kitas.mtalpa_)
00076 {
00077     kitas.mduom_ = nullptr;
00078     kitas.mdydis_ = 0;
00079     kitas.mtalpa_ = 0;
00080 }
00081
00082 // (6) Initializer list constructor
00083 Vektorius(std::initializer_list<T> sarasas)
00084     : mduom_(new T[sarasas.size()]), mdydis_(sarasas.size()), mtalpa_(sarasas.size())
00085 {
00086     std::copy(sarasas.begin(), sarasas.end(), mduom_);
00087 }
00088
00089
00090
00091 // Dekstruktorius
00092 ~Vektorius()
00093 {
00094     delete[] mduom_;
00095 }
00096 // Operatoriai =
00097 // Kopijavimo priskyrimo operatorius
00098 Vektorius& operator=(const Vektorius& kitas)
00099 {
00100     if (this != &kitas) {
00101         T* naujiDuom = new T[kitas.mdydis_];
00102         for(int i = 0; i !=kitas.mdydis_; ++i)
00103         {
00104             naujiDuom[i]=kitas.mduom_[i];
00105         }
00106         delete[] mduom_; // atlaisvinama sena atmintis
00107         mduom_ = naujiDuom; // mduom pointina i nauja atminti
00108         mdydis_ = kitas.mdydis_;
00109         mtalpa_ = kitas.mtalpa_;
00110     }
00111     return *this;
00112 }
00113 // Perkelimo priskyrimo operatorius
00114 Vektorius& operator=(Vektorius&& kitas) { // pavogiame objekto duomenys priskyrimo metu
00115     // Savęs priskyrimo aptikimas
00116     if (&kitas == this) return *this;
00117     delete[] mduom_; // atlaisviname seną atmintį
00118     mduom_ = kitas.mduom_; // elem point'ina į v.elem atmintį
00119     mdydis_ = kitas.mdydis_; // atnaujiname size
00120     mtalpa_ = kitas.mtalpa_;
00121     kitas.mduom_ = nullptr; // v neturi jokių elementų
00122     kitas.mdydis_ = 0;
00123     kitas.mtalpa_ = 0;
00124
00125     return *this; // grąžiname objektą
00126 }
00127
00128
00129 // element access
00130
00131 T& operator[](size_type indeksas)
00132 {
00133     if (indeksas >= mdydis_)
00134         throw std::out_of_range("Index out of range");
00135     return mduom_[indeksas];
00136 }
00137
00138 const T& operator[](size_type indeksas) const
00139 {
00140     if (indeksas >= mdydis_)
00141         throw std::out_of_range("Index out of range");
00142     return mduom_[indeksas];
00143 }
00144
00145 T& at(size_type indeksas)
00146 {
00147     if (indeksas >= mdydis_)
00148         throw std::out_of_range("Index out of range");
00149     return mduom_[indeksas];
00150 }
00151
00152 const T& at(size_type indeksas) const
00153 {
00154     if (indeksas >= mdydis_)
00155         throw std::out_of_range("Index out of range");

```

```

00156         return mduom_[indeksas];
00157     }
00158     T& front()
00159     {
00160         if (mdydis_ == 0)
00161             throw std::out_of_range("Vektorius tuscias");
00162         return mduom_[0];
00163     }
00164
00165     const T& front() const
00166     {
00167         if (mdydis_ == 0)
00168             throw std::out_of_range("Vektorius tuscias");
00169         return mduom_[0];
00170     }
00171
00172     T& back()
00173     {
00174         if (mdydis_ == 0)
00175             throw std::out_of_range("Vektorius tuscias");
00176         return mduom_[mdydis_ - 1];
00177     }
00178
00179     const T& back() const
00180     {
00181         if (mdydis_ == 0)
00182             throw std::out_of_range("Vektorius tuscias");
00183         return mduom_[mdydis_ - 1];
00184     }
00185     T* data() noexcept { return mduom_; }
00186
00187     const T* data() const noexcept { return mduom_; }
00188     // capacity
00189
00190     size_type size() const noexcept { return mdydis_; }
00191     size_type capacity() const noexcept { return mtalpa_; }
00192     bool empty() const noexcept { return mdydis_ == 0; }
00193     size_type max_size() const noexcept { return std::numeric_limits<size_type>::max(); }
00194     void reserve(size_type naujaTalpa)
00195     {
00196         if (naujaTalpa <= mtalpa_)
00197             return;
00198
00199         T* naujiDuum = new T[naujaTalpa];
00200         for (size_type k = 0; k < mdydis_; ++k)
00201             naujiDuum[k] = std::move(mduom_[k]);
00202
00203         delete[] mduom_;
00204         mduom_ = naujiDuum;
00205         mtalpa_ = naujaTalpa;
00206     }
00207     void shrink_to_fit()
00208     {
00209         if (mdydis_ < mtalpa_) {
00210             T* naujiDuum = new T[mdydis_];
00211             std::copy(mduom_, mduom_ + mdydis_, naujiDuum);
00212             delete[] mduom_;
00213             mduom_ = naujiDuum;
00214             mtalpa_ = mdydis_;
00215         }
00216     }
00217
00218
00219
00220
00221
00222     // ITERATORIAI
00223     iterator begin() noexcept { return mduom_; }
00224     const_iterator begin() const noexcept { return mduom_; }
00225     iterator end() noexcept { return mduom_ + mdydis_; }
00226     const_iterator end() const noexcept { return mduom_ + mdydis_; }
00227     const_iterator cbegin() const noexcept { return mduom_; }
00228     const_iterator cend() const noexcept { return mduom_ + mdydis_; }
00229     reverse_iterator rbegin() noexcept { return reverse_iterator(end()); }
00230     const_reverse_iterator rbegin() const noexcept { return const_reverse_iterator(end()); }
00231     reverse_iterator rend() noexcept { return reverse_iterator(begin()); }
00232     const_reverse_iterator rend() const noexcept { return const_reverse_iterator(begin()); }
00233     const_reverse_iterator crbegin() const noexcept { return const_reverse_iterator(end()); }
00234     const_reverse_iterator crend() const noexcept { return const_reverse_iterator(begin()); }
00235
00236     // MODIFIERS
00237
00238     void clear() noexcept
00239     {
00240         mdydis_ = 0;
00241     }
00242     iterator insert(const_iterator pos, const T& value) {

```

```

00243         size_type index = pos - begin();
00244     if (mdydis_ == mtaalpa_) {
00245         size_type new_capacity = (mtaalpa_ == 0) ? 1 : mtaalpa_ * 2;
00246         reserve(new_capacity);
00247     }
00248
00249     // Perstumiam visus elementus nuo ėterpimo vietos į dešinę per vieną
00250     for (size_type i = mdydis_; i > index; --i) {
00251         mduom_[i] = std::move(mduom_[i - 1]);
00252     }
00253
00254     // Įterpiame naują elementą į vietą 'pos'
00255     mduom_[index] = value;
00256     ++mdydis_;
00257
00258     return begin() + index;
00259 }
00260 iterator erase(const_iterator pos) {
00261     size_type index = pos - begin();
00262     if (index >= mdydis_) {
00263         throw std::out_of_range("Index out of range");
00264     }
00265
00266     // Perstumiam visus elementus nuo 'pos' vienetu į kairę
00267     for (size_type i = index; i < mdydis_ - 1; ++i) {
00268         mduom_[i] = std::move(mduom_[i + 1]);
00269     }
00270
00271     --mdydis_;
00272     return begin() + index;
00273 }
00274 iterator erase(const_iterator first, const_iterator last)
00275 {
00276     size_type first_index = first - begin();
00277     size_type last_index = last - begin();
00278     if (first_index > last_index || last_index > mdydis_) {
00279         throw std::out_of_range("Invalid range");
00280     }
00281
00282     size_type num_to_erase = last_index - first_index;
00283     for (size_type i = first_index; i < mdydis_ - num_to_erase; ++i) {
00284         mduom_[i] = std::move(mduom_[i + num_to_erase]);
00285     }
00286
00287     mdydis_ -= num_to_erase;
00288     return begin() + first_index;
00289 }
00290 void push_back(const T& x)
00291 {
00292     if (mdydis_ == mtaalpa_)
00293         reserve(mtaalpa_ == 0 ? 1 : mtaalpa_ * 2);
00294     mduom_[mdydis_++] = x;
00295 }
00296
00297
00298
00299 void pop_back()
00300 {
00301     if (mdydis_ > 0)
00302         --mdydis_;
00303 }
00304 // void resize(size_type new_size, const T& value = T())
00305 // {
00306 //     if (new_size < mdydis_) {
00307 //         mdydis_ = new_size;
00308 //     } else if (new_size > mdydis_) {
00309 //         reserve(new_size);
00310 //         for (size_type i = mdydis_; i < new_size; ++i) {
00311 //             mduom_[i] = value;
00312 //         }
00313 //         mdydis_ = new_size;
00314 //     }
00315 // }
00316 void resize(size_type new_size)
00317 {
00318     if (new_size < mdydis_) {
00319         mdydis_ = new_size;
00320     } else if (new_size > mdydis_) {
00321         reserve(new_size);
00322         mdydis_ = new_size;
00323     }
00324 }
00325 void swap(Vektorius& other) noexcept
00326 {
00327     //using std::swap; // Importuojame swap iš std
00328
00329     // Keičiame visus narius su kitu vektoriumi

```



```

00330         swap(mduom_, other.mduom_);
00331         swap(mdydis_, other.mdydis_);
00332         swap(mtalpa_, other.mtalpa_);
00333     }
00334     // NON-MEMBER FUNCTIONS
00335     bool operator== (const Vektorius<T>& other) const {
00336         if (size() != other.size()) {
00337             return false;
00338         }
00339         return std::equal(begin(), end(), other.begin());
00340     }
00341     bool operator!= (const Vektorius<T>& other) const {
00342         return !(*this == other);
00343     }
00344     bool operator < (const Vektorius<T> & other) const {
00345         return std::lexicographical_compare(begin(), end(), other.begin(), other.end());
00346     }
00347     bool operator <= (const Vektorius<T> & other) const {
00348         return !(other < *this);
00349     }
00350     bool operator > (const Vektorius<T> & other) const {
00351         return std::lexicographical_compare(other.begin(), other.end(), begin(), end());
00352     }
00353     bool operator >= (const Vektorius<T> & other) const {
00354         return !(other > *this);
00355     }
00356     void swap (Vektorius<T>& x, Vektorius<T>& y) {
00357         std::swap(x,y);
00358     }
00359     void print() const {
00360         std::cout << "{";
00361         for(size_type i = 0; i < mdydis_; ++i) {
00362             std::cout << mduom_[i] << " ";
00363         }
00364         std::cout << "}" << std::endl;
00365     }
00366 };
00367 #endif /* vektorius_h */

```



# Index

- ~Vartotojas
  - Vartotojas, [9](#)
- ~Vektorius
  - Vektorius< T >, [16](#)
- ~Zmogus
  - Zmogus, [23](#)
- arSveikasisSk
  - funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- arZodis
  - funkcijos.cpp, [27](#)
  - funkcijos.h, [31](#)
- at
  - Vektorius< T >, [16](#)
- back
  - Vektorius< T >, [16](#)
- begin
  - Vektorius< T >, [16](#)
- biblioteka.h, [25](#)
- capacity
  - Vektorius< T >, [16](#)
- cbegin
  - Vektorius< T >, [17](#)
- cend
  - Vektorius< T >, [17](#)
- clear
  - Vektorius< T >, [17](#)
- const\_iterator
  - Vektorius< T >, [14](#)
- const\_reverse\_iterator
  - Vektorius< T >, [14](#)
- crbegin
  - Vektorius< T >, [17](#)
- crend
  - Vektorius< T >, [17](#)
- data
  - Vektorius< T >, [17](#)
- empty
  - Vektorius< T >, [17](#)
- end
  - Vektorius< T >, [17](#), [18](#)
- erase
  - Vektorius< T >, [18](#)
- FailuGeneravimas
  - funkcijos.cpp, [27](#)
- funkcijos.h, [31](#)
- front
  - Vektorius< T >, [18](#)
- funkcijos.cpp, [26](#)
  - arSveikasisSk, [27](#)
  - arZodis, [27](#)
  - FailuGeneravimas, [27](#)
  - generavimasPav, [27](#)
  - generavimasPaz, [27](#)
  - generavimasVard, [27](#)
  - Mediana, [28](#)
  - rezrikiavimas, [28](#)
  - rikiuotiMed, [28](#)
  - rikiuotiPavarde, [28](#)
  - rikiuotiVarda, [28](#)
  - rikiuotiVid, [28](#)
  - RusiavimasDviGrupes, [28](#)
  - RusiavimasDviGrupes2, [28](#)
  - RusiavimasDviGrupes3, [29](#)
  - skaityti, [29](#)
  - spausdinti, [29](#)
  - spausdinti\_skaitomus\_duomenis, [29](#)
  - spausdintiLaimingiVargsai, [29](#)
  - testas, [29](#)
  - Vidurkis, [29](#)
- funkcijos.h, [30](#)
  - arSveikasisSk, [31](#)
  - arZodis, [31](#)
  - FailuGeneravimas, [31](#)
  - generavimasPav, [31](#)
  - generavimasPaz, [31](#)
  - generavimasVard, [31](#)
  - Mediana, [31](#)
  - rezrikiavimas, [31](#)
  - rikiuotiMed, [31](#)
  - rikiuotiPavarde, [32](#)
  - rikiuotiVarda, [32](#)
  - rikiuotiVid, [32](#)
  - RusiavimasDviGrupes, [32](#)
  - RusiavimasDviGrupes2, [32](#)
  - RusiavimasDviGrupes3, [32](#)
  - skaityti, [32](#)
  - spausdinti, [33](#)
  - spausdinti\_skaitomus\_duomenis, [33](#)
  - spausdintiLaimingiVargsai, [33](#)
  - testas, [33](#)
  - Vidurkis, [33](#)
- generavimasPav
  - funkcijos.cpp, [27](#)

- funkcijos.h, 31
- generavimasPaz
  - funkcijos.cpp, 27
  - funkcijos.h, 31
- generavimasVard
  - funkcijos.cpp, 27
  - funkcijos.h, 31
- getEgz
  - Vartotojas, 10
- getGal
  - Vartotojas, 10
- getGalmed
  - Vartotojas, 10
- getGalvid
  - Vartotojas, 10
- getMed
  - Vartotojas, 10
- getPav
  - Vartotojas, 10
  - Zmogus, 23
- getPaz
  - Vartotojas, 10
- getVar
  - Vartotojas, 11
  - Zmogus, 23
- getVid
  - Vartotojas, 11
- insert
  - Vektorius< T >, 18
- iterator
  - Vektorius< T >, 14
- main
  - v3.cpp, 36
- max\_size
  - Vektorius< T >, 18
- Mediana
  - funkcijos.cpp, 28
  - funkcijos.h, 31
- operator!=
  - Vektorius< T >, 18
- operator<
  - Vektorius< T >, 19
- operator<<
  - Vartotojas, 13
- operator<=
  - Vektorius< T >, 19
- operator>
  - Vektorius< T >, 19
- operator>>
  - Vartotojas, 13
- operator>=
  - Vektorius< T >, 19
- operator=
  - Vartotojas, 11
  - Vektorius< T >, 19
- operator==
  - Vektorius< T >, 19
- operator[]
  - Vektorius< T >, 19, 20
- pavarde\_
  - Zmogus, 23
- pop\_back
  - Vektorius< T >, 20
- print
  - Vektorius< T >, 20
- push\_back
  - Vektorius< T >, 20
- rbegin
  - Vektorius< T >, 20
- rend
  - Vektorius< T >, 20
- reserve
  - Vektorius< T >, 21
- resize
  - Vektorius< T >, 21
- reverse\_iterator
  - Vektorius< T >, 15
- rezrikiavimas
  - funkcijos.cpp, 28
  - funkcijos.h, 31
- rikiuotiMed
  - funkcijos.cpp, 28
  - funkcijos.h, 31
- rikiuotiPavarde
  - funkcijos.cpp, 28
  - funkcijos.h, 32
- rikiuotiVarda
  - funkcijos.cpp, 28
  - funkcijos.h, 32
- rikiuotiVid
  - funkcijos.cpp, 28
  - funkcijos.h, 32
- RusiavimasDviGrupes
  - funkcijos.cpp, 28
  - funkcijos.h, 32
- RusiavimasDviGrupes2
  - funkcijos.cpp, 28
  - funkcijos.h, 32
- RusiavimasDviGrupes3
  - funkcijos.cpp, 29
  - funkcijos.h, 32
- setEgz
  - Vartotojas, 11
- setGal
  - Vartotojas, 11
- setGalmed
  - Vartotojas, 11
- setGalvid
  - Vartotojas, 12
- setMed
  - Vartotojas, 12
- setPav

- Vartotojas, 12
- Zmogus, 23
- setPaz
  - Vartotojas, 12
- setVar
  - Vartotojas, 12
  - Zmogus, 23
- setVid
  - Vartotojas, 12
- shrink\_to\_fit
  - Vektorius< T >, 21
- size
  - Vektorius< T >, 21
- size\_type
  - Vektorius< T >, 15
- skaityti
  - funkcijos.cpp, 29
  - funkcijos.h, 32
- spausdinti
  - funkcijos.cpp, 29
  - funkcijos.h, 33
- spausdinti\_skaitomus\_duomenis
  - funkcijos.cpp, 29
  - funkcijos.h, 33
- spausdintiLaimingiVargsai
  - funkcijos.cpp, 29
  - funkcijos.h, 33
- swap
  - Vektorius< T >, 21
- testas
  - funkcijos.cpp, 29
  - funkcijos.h, 33
- v3.cpp, 36
  - main, 36
- value\_type
  - Vektorius< T >, 15
- vardas\_
  - Zmogus, 23
- Vartotojas, 7
  - ~Vartotojas, 9
  - getEgz, 10
  - getGal, 10
  - getGalmed, 10
  - getGalvid, 10
  - getMed, 10
  - getPav, 10
  - getPaz, 10
  - getVar, 11
  - getVid, 11
  - operator<<, 13
  - operator>>, 13
  - operator=, 11
  - setEgz, 11
  - setGal, 11
  - setGalmed, 11
  - setGalvid, 12
  - setMed, 12
- setPav, 12
- setPaz, 12
- setVar, 12
- setVid, 12
- Vartotojas, 9
- Vektorius
  - Vektorius< T >, 15
- Vektorius< T >, 13
  - ~Vektorius, 16
  - at, 16
  - back, 16
  - begin, 16
  - capacity, 16
  - cbegin, 17
  - cend, 17
  - clear, 17
  - const\_iterator, 14
  - const\_reverse\_iterator, 14
  - crbegin, 17
  - crend, 17
  - data, 17
  - empty, 17
  - end, 17, 18
  - erase, 18
  - front, 18
  - insert, 18
  - iterator, 14
  - max\_size, 18
  - operator!=, 18
  - operator<, 19
  - operator<=, 19
  - operator>, 19
  - operator>=, 19
  - operator=, 19
  - operator==, 19
  - operator[], 19, 20
  - pop\_back, 20
  - print, 20
  - push\_back, 20
  - rbegin, 20
  - rend, 20
  - reserve, 21
  - resize, 21
  - reverse\_iterator, 15
  - shrink\_to\_fit, 21
  - size, 21
  - size\_type, 15
  - swap, 21
  - value\_type, 15
  - Vektorius, 15
- vektorius.h, 36
- Vidurkis
  - funkcijos.cpp, 29
  - funkcijos.h, 33
- Zmogus, 22
  - ~Zmogus, 23
  - getPav, 23
  - getVar, 23

pavarde\_, [23](#)  
setPav, [23](#)  
setVar, [23](#)  
vardas\_, [23](#)  
Zmogus, [22](#)