

# Objektinio Programavimo 3 Užduotis

## V3.0

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 ManoVector< T > Class Template Reference	7
4.1.1 Detailed Description	9
4.1.2 Constructor & Destructor Documentation	9
4.1.2.1 ManoVector() [1/4]	9
4.1.2.2 ManoVector() [2/4]	9
4.1.2.3 ManoVector() [3/4]	9
4.1.2.4 ManoVector() [4/4]	9
4.1.2.5 ~ManoVector()	10
4.1.3 Member Function Documentation	10
4.1.3.1 assign()	10
4.1.3.2 at() [1/2]	10
4.1.3.3 at() [2/2]	10
4.1.3.4 back() [1/2]	11
4.1.3.5 back() [2/2]	11
4.1.3.6 begin() [1/2]	11
4.1.3.7 begin() [2/2]	12
4.1.3.8 capacity()	12
4.1.3.9 clear()	12
4.1.3.10 data()	12
4.1.3.11 emplace()	12
4.1.3.12 emplace_back()	13
4.1.3.13 empty()	13
4.1.3.14 end() [1/2]	13
4.1.3.15 end() [2/2]	13
4.1.3.16 erase() [1/2]	14
4.1.3.17 erase() [2/2]	14
4.1.3.18 front() [1/2]	14
4.1.3.19 front() [2/2]	14
4.1.3.20 insert()	15
4.1.3.21 max_size()	15
4.1.3.22 operator=() [1/2]	15
4.1.3.23 operator=() [2/2]	15
4.1.3.24 operator[]() [1/2]	16

4.1.3.25 operator[]() [2/2]	16
4.1.3.26 pop_back()	16
4.1.3.27 push_back()	16
4.1.3.28 reserve()	16
4.1.3.29 resize()	17
4.1.3.30 shrink_to_fit()	17
4.1.3.31 size()	17
4.1.3.32 swap()	17
4.2 Studentas Class Reference	18
4.2.1 Detailed Description	20
4.2.2 Constructor & Destructor Documentation	20
4.2.2.1 Studentas() [1/4]	20
4.2.2.2 Studentas() [2/4]	20
4.2.2.3 ~Studentas()	20
4.2.2.4 Studentas() [3/4]	20
4.2.2.5 Studentas() [4/4]	21
4.2.3 Member Function Documentation	21
4.2.3.1 ClearEverything()	21
4.2.3.2 DeleteLastNd()	21
4.2.3.3 Get_Egzaminas()	21
4.2.3.4 Get_Last_Nd()	21
4.2.3.5 Get_Mediana()	22
4.2.3.6 Get_Nd()	22
4.2.3.7 Get_Vidurkis()	22
4.2.3.8 medianosSkaiciavimas()	22
4.2.3.9 ND_clear()	23
4.2.3.10 Nd_dydis()	23
4.2.3.11 Nd_empty()	23
4.2.3.12 nd_rusiavimas()	23
4.2.3.13 Nd_Suma()	23
4.2.3.14 operator=() [1/2]	23
4.2.3.15 operator=() [2/2]	24
4.2.3.16 Print()	24
4.2.3.17 setEgzaminas()	24
4.2.3.18 SetMediana()	24
4.2.3.19 setNd()	25
4.2.3.20 setVidurkis()	25
4.2.3.21 Vidurkis()	25
4.2.4 Friends And Related Symbol Documentation	25
4.2.4.1 operator<< [1/2]	25
4.2.4.2 operator<< [2/2]	26
4.2.4.3 operator>> [1/2]	26

4.2.4.4 operator>> [2/2]	26
4.3 Zmogus Class Reference	27
4.3.1 Detailed Description	28
4.3.2 Constructor & Destructor Documentation	28
4.3.2.1 Zmogus() [1/2]	28
4.3.2.2 Zmogus() [2/2]	28
4.3.2.3 ~Zmogus()	28
4.3.3 Member Function Documentation	28
4.3.3.1 Get_Pavarde()	28
4.3.3.2 Get_Vardas()	29
4.3.3.3 Print()	29
4.3.3.4 SetPavarde()	29
4.3.3.5 SetVardas()	29
4.3.4 Member Data Documentation	30
4.3.4.1 pavarde	30
4.3.4.2 vardas	30
<b>5 File Documentation</b>	<b>31</b>
5.1 Headers/funkcijos.h File Reference	31
5.1.1 Function Documentation	32
5.1.1.1 GeneruotiFaila()	32
5.1.1.2 GeneruotiPazymius()	32
5.1.1.3 GeneruotiStudenta()	33
5.1.1.4 IsvestiRezultatus()	33
5.1.1.5 palygintiPagalMediana()	33
5.1.1.6 palygintiPagalPavarde()	33
5.1.1.7 palygintiPagalVarda()	35
5.1.1.8 palygintiPagalVidurki()	35
5.1.1.9 PasalintiKietusStudentus()	35
5.1.2 Variable Documentation	36
5.1.2.1 choice3	36
5.1.2.2 Lievi	36
5.1.2.3 norima_isvedimo_vieta	36
5.1.2.4 norimas_rikiavimas	36
5.1.2.5 Pavardes	36
5.1.2.6 programos_tesinys	36
5.1.2.7 programos_veikimas	37
5.1.2.8 Studentai	37
5.1.2.9 Vardai	37
5.2 funkcijos.h	37
5.3 Headers/Vector.h File Reference	38
5.4 Vector.h	39

5.5 Headers/Zmogus.h File Reference	41
5.6 Zmogus.h	42
5.7 Sources/funkcijos.cpp File Reference	42
5.7.1 Function Documentation	43
5.7.1.1 GeneruotiFaila()	43
5.7.1.2 GeneruotiPazymius()	43
5.7.1.3 GeneruotiStudenta()	43
5.7.1.4 IvestiRezultatus()	43
5.7.1.5 operator<<() [1/2]	44
5.7.1.6 operator<<() [2/2]	44
5.7.1.7 operator>>() [1/2]	44
5.7.1.8 operator>>() [2/2]	45
5.7.1.9 palygintiPagalMediana()	45
5.7.1.10 palygintiPagalPavarde()	45
5.7.1.11 palygintiPagalVarda()	46
5.7.1.12 palygintiPagalVidurki()	46
5.7.1.13 PasalintiKietusStudentus()	46
5.8 Sources/Vector_v2_0.cpp File Reference	47
5.8.1 Function Documentation	47
5.8.1.1 main()	47
5.8.2 Variable Documentation	47
5.8.2.1 N	47
5.8.2.2 programos_veikimas	47
5.9 Testavimas/Testavimas.cpp File Reference	48
5.9.1 Function Documentation	48
5.9.1.1 main()	48
5.9.1.2 TEST() [1/7]	48
5.9.1.3 TEST() [2/7]	48
5.9.1.4 TEST() [3/7]	48
5.9.1.5 TEST() [4/7]	49
5.9.1.6 TEST() [5/7]	49
5.9.1.7 TEST() [6/7]	49
5.9.1.8 TEST() [7/7]	49
5.9.2 Variable Documentation	49
5.9.2.1 programos_veikimas	49
5.10 Testavimas/Vektoriaus_Testavimas.cpp File Reference	49
5.10.1 Function Documentation	50
5.10.1.1 main()	50
5.10.1.2 TEST() [1/29]	50
5.10.1.3 TEST() [2/29]	50
5.10.1.4 TEST() [3/29]	51
5.10.1.5 TEST() [4/29]	51

5.10.1.6 TEST() [5/29]	51
5.10.1.7 TEST() [6/29]	51
5.10.1.8 TEST() [7/29]	51
5.10.1.9 TEST() [8/29]	51
5.10.1.10 TEST() [9/29]	51
5.10.1.11 TEST() [10/29]	51
5.10.1.12 TEST() [11/29]	52
5.10.1.13 TEST() [12/29]	52
5.10.1.14 TEST() [13/29]	52
5.10.1.15 TEST() [14/29]	52
5.10.1.16 TEST() [15/29]	52
5.10.1.17 TEST() [16/29]	52
5.10.1.18 TEST() [17/29]	52
5.10.1.19 TEST() [18/29]	52
5.10.1.20 TEST() [19/29]	53
5.10.1.21 TEST() [20/29]	53
5.10.1.22 TEST() [21/29]	53
5.10.1.23 TEST() [22/29]	53
5.10.1.24 TEST() [23/29]	53
5.10.1.25 TEST() [24/29]	53
5.10.1.26 TEST() [25/29]	53
5.10.1.27 TEST() [26/29]	53
5.10.1.28 TEST() [27/29]	54
5.10.1.29 TEST() [28/29]	54
5.10.1.30 TEST() [29/29]	54

<b>Index</b>	<b>55</b>
--------------	-----------





# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

ManoVector< T > . . . . .	7
ManoVector< int > . . . . .	7
Zmogus . . . . .	27
Studentas . . . . .	18



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">ManoVector&lt; T &gt;</a>	Vektoriaus klasė pritaikyta šitai programai . . . . .	7
<a href="#">Studentas</a>	Studento klasė . . . . .	18
<a href="#">Zmogus</a>	Abstrakti žmogaus klasė . . . . .	27



# Chapter 3

## File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

Headers/ <a href="#">funkcijos.h</a> . . . . .	31
Headers/ <a href="#">Vector.h</a> . . . . .	38
Headers/ <a href="#">Zmogus.h</a> . . . . .	41
Sources/ <a href="#">funkcijos.cpp</a> . . . . .	42
Sources/ <a href="#">Vector_v2_0.cpp</a> . . . . .	47
Testavimas/ <a href="#">Testavimas.cpp</a> . . . . .	48
Testavimas/ <a href="#">Vektoriaus_Testavimas.cpp</a> . . . . .	49



## Chapter 4

# Class Documentation

### 4.1 `ManoVector< T >` Class Template Reference

Vektoriaus klasė pritaikyta šitai programai.

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

#### Public Member Functions

- `int max_size () const`  
*Funkcija, kuri grąžina maksimalų galimą vektoriaus dydį.*
- `ManoVector ()`  
*Default'inis konstruktorius.*
- `ManoVector (std::initializer_list< T > il)`  
*initializer\_list konstruktorius*
- `ManoVector (const ManoVector< T > &Kitas_Vektorius)`  
*Kopijavimo konstruktorius.*
- `ManoVector (ManoVector &&Kitas_Vektorius) noexcept`  
*Move konstruktorius.*
- `~ManoVector ()`  
*ManoVector destruktoriaus.*
- `ManoVector & operator= (const ManoVector &Kitas_Vektorius)`  
*Kopijavimo operatorius.*
- `ManoVector & operator= (ManoVector &&Kitas_Vektorius)`  
*Move operatorius.*
- `T & operator[] (unsigned int indeksas)`  
*Operatorius, kuris grąžina nurodytą elementą.*
- `const T & operator[] (unsigned int indeksas) const`  
*Operatorius, kuris grąžina nurodytą elementą.*
- `T & at (unsigned int indeksas)`  
*Funkcija, kuri grąžina nurodytą elementą.*
- `const T & at (unsigned int indeksas) const`  
*Funkcija, kuri grąžina nurodytą elementą.*
- `T & front ()`  
*Funkcija, kuri grąžina pirmą elementą.*

- `const T & front () const`  
*Funkcija, kuri grąžina pirmą elementą.*
- `T & back ()`  
*Funkcija, kuri grąžina paskutinį elementą.*
- `const T & back () const`  
*Funkcija, kuri grąžina paskutinį elementą.*
- `T * data () noexcept`  
*Funkcija, kuri grąžina rodyklę į duomenis.*
- `T * begin () noexcept`  
*Funkcija, kuri grąžina rodyklę į pirmą vektoriaus elementą.*
- `const T * begin () const noexcept`  
*Funkcija, kuri grąžina konstantinę rodyklę į pirmą vektoriaus elementą.*
- `T * end () noexcept`  
*Funkcija, kuri grąžina rodyklę į vektoriaus pabaigą.*
- `const T * end () const noexcept`  
*Funkcija, kuri grąžina konstantinę rodyklę į vektoriaus pabaigą.*
- `unsigned int capacity () const`  
*Funkcija, kuri grąžina vektoriaus talpą.*
- `unsigned int size () const`  
*Funkcija, kuri grąžina vektoriaus dydį.*
- `bool empty () const`  
*Funkcija, kuri patikrina ar vektorius tuščias.*
- `void reserve (unsigned int nauja_talpa)`  
*Funkcija, kuri rezervuoja vietas vektoriuje.*
- `void shrink_to_fit ()`  
*Funkcija, kuri sumažina vektoriaus talpą iki vektoriaus dydžio.*
- `void assign (unsigned int n, const T &value)`  
*Funkcija, kuri priskiria nurodytą elementą nurodytam kiekiui.*
- `void clear ()`  
*Funkcija, išvalanti visą vektorių*
- `void push_back (const T &value)`  
*Funkcija, kuri prideda elementą į vektoriaus pabaigą.*
- `T * insert (unsigned int indeksas, const T &value)`  
*Funkcija, kuri prideda elementą į nurodytą poziciją.*
- `T * emplace (unsigned int indeksas, T &&value)`  
*Funkcija, kuri prideda elementą į nurodytą poziciją.*
- `T & emplace_back (T &&value)`  
*Funkcija, kuri prideda elementą į vektoriaus pabaigą.*
- `void pop_back ()`  
*Funkcija, kuri pašalina elementą iš **ManoVector** pabaigos.*
- `void resize (unsigned int naujas_dydis)`  
*Funkcija, kuri pakeičia **ManoVector** dydį į nurodytą dydį.*
- `void swap (ManoVector &Kitas_Vektorius)`  
*Funkcija, kuri sukeičia du vektorių vietomis.*
- `T * erase (unsigned int indeksas)`  
*Funkcija, kuri išmeta nurodytą elementą.*
- `T * erase (T *pirmas_elementas, T *paskutinis_elementas)`  
*Funkcija, kuri išmeta elementus tarp nurodytų elementų.*



### 4.1.1 Detailed Description

```
template<typename T>
class ManoVector< T >
```

Vektoriaus klasė pritaikyta šitai programai.

### 4.1.2 Constructor & Destructor Documentation

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

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

Default'inis konstruktorius.

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

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

initializer\_list konstruktorius

#### 4.1.2.3 ManoVector() [3/4]

```
template<typename T >
ManoVector< T >::ManoVector (
    const ManoVector< T > & Kitas_Vektorius ) [inline]
```

Kopijavimo konstruktorius.

##### Parameters

<i>Kitas_Vektorius</i>	Vektorius, iš kurio kopijuojami duomenys.
------------------------	---

#### 4.1.2.4 ManoVector() [4/4]

```
template<typename T >
ManoVector< T >::ManoVector (
    ManoVector< T > && Kitas_Vektorius ) [inline], [noexcept]
```

Move konstruktorius.

##### Parameters

<i>Kitas_Vektorius</i>	Vektorius, iš kurio perkeliama duomenys.
------------------------	--

#### 4.1.2.5 ~ManoVector()

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

[ManoVector](#) destruktoius.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 assign()

```
template<typename T >
void ManoVector< T >::assign (
    unsigned int n,
    const T & value ) [inline]
```

Funkcija, kuri priskiria nurodytą elementą nurodytam kiekiui.

##### Parameters

<i>n</i>	Kiekis.
<i>value</i>	Elementas.

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

```
template<typename T >
T & ManoVector< T >::at (
    unsigned int indeksas ) [inline]
```

Funkcija, kuri grąžina nurodytą elementą.

##### Parameters

<i>indeksas</i>	Elemento indeksas.
-----------------	--------------------

##### Returns

Elementas.

##### Exceptions

<i>std::out_of_range</i>	Jei indeksas už ribų.
--------------------------	-----------------------

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

```
template<typename T >
```

```
const T & ManoVector< T >::at (
    unsigned int indeksas ) const [inline]
```

Funkcija, kuri grąžina nurodytą elementą.

#### Parameters

<i>indeksas</i>	Elemento indeksas.
-----------------	--------------------

#### Returns

Elementas.

#### Exceptions

<i>std::out_of_range</i>	Jei indeksas už ribų.
--------------------------	-----------------------

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

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

Funkcija, kuri grąžina paskutinį elementą.

#### Returns

Paskutinis elementas.

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

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

Funkcija, kuri grąžina paskutinį elementą.

#### Returns

Paskutinis elementas.

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

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

Funkcija, kuri grąžina konstantinę rodyklę į pirmą vektoriaus elementą.

#### Returns

Konstantinė rodyklė į pirmą vektoriaus elementą.

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

```
template<typename T >
T * ManoVector< T >::begin ( ) [inline], [noexcept]
```

Funkcija, kuri grąžina rodyklę į pirmą vektoriaus elementą.

##### Returns

Rodyklė į pirmą vektoriaus elementą.

#### 4.1.3.8 capacity()

```
template<typename T >
unsigned int ManoVector< T >::capacity ( ) const [inline]
```

Funkcija, kuri grąžina vektoriaus talpą.

##### Returns

Vektoriaus talpa.

#### 4.1.3.9 clear()

```
template<typename T >
void ManoVector< T >::clear ( ) [inline]
```

Funkcija, išvalanti visą vektorių

#### 4.1.3.10 data()

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

Funkcija, kuri grąžina rodyklę į duomenis.

##### Returns

Rodyklė į duomenis.

#### 4.1.3.11 emplace()

```
template<typename T >
T * ManoVector< T >::emplace (
    unsigned int indeksas,
    T && value ) [inline]
```

Funkcija, kuri prideda elementą į nurodytą poziciją.

## Parameters

<i>value</i>	Elementas, kuris turi būti pridėtas.
--------------	--------------------------------------

**4.1.3.12** `emplace_back()`

```
template<typename T >
T & ManoVector< T >::emplace_back (
    T && value ) [inline]
```

Funkcija, kuri prideda elementą į vektoriaus pabaigą.

## Parameters

<i>value</i>	Elementas, kuris turi būti pridėtas.
--------------	--------------------------------------

**4.1.3.13** `empty()`

```
template<typename T >
bool ManoVector< T >::empty ( ) const [inline]
```

Funkcija, kuri patikrina ar vektorius tuščias.

## Returns

Ar vektorius tuščias.

**4.1.3.14** `end()` [1/2]

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

Funkcija, kuri grąžina konstantinę rodyklę į vektoriaus pabaigą.

## Returns

Konstantinė rodyklė į vektoriaus pabaigą.

**4.1.3.15** `end()` [2/2]

```
template<typename T >
T * ManoVector< T >::end ( ) [inline], [noexcept]
```

Funkcija, kuri grąžina rodyklę į vektoriaus pabaigą.

## Returns

Rodyklė į vektoriaus pabaigą.

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

```
template<typename T >
T * ManoVector< T >::erase (
    T * pirmas_elementas,
    T * paskutinis_elementas ) [inline]
```

Funkcija, kuri išmeta elementus tarp nurodytų elementų.

**Parameters**

<i>pirmas_elementas</i>	Pirmas elementas.
<i>paskutinis_elementas</i>	Paskutinis elementas.

**Returns**

Rodyklė į paskutinį elementą.

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

```
template<typename T >
T * ManoVector< T >::erase (
    unsigned int indeksas ) [inline]
```

Funkcija, kuri išmeta nurodytą elementą.

**Parameters**

<i>indeksas</i>	Elemento indeksas.
-----------------	--------------------

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

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

Funkcija, kuri grąžina pirmą elementą.

**Returns**

Pirmas elementas.

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

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

Funkcija, kuri grąžina pirmą elementą.

**Returns**

Pirmas elementas.

#### 4.1.3.20 insert()

```
template<typename T >
T * ManoVector< T >::insert (
    unsigned int indeksas,
    const T & value ) [inline]
```

Funkcija, kuri prideda elementą į nurodytą poziciją.

##### Parameters

<i>value</i>	Elementas, kuris turi būti pridėtas.
--------------	--------------------------------------

#### 4.1.3.21 max\_size()

```
template<typename T >
int ManoVector< T >::max_size ( ) const [inline]
```

Funkcija, kuri grąžina maksimalų galimą vektoriaus dydį.

##### Returns

Maksimalus vektoriaus dydis.

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

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

Kopijavimo operatorius.

##### Parameters

<i>Kitas_Vektorius</i>	Vektorius, iš kurio kopijuojami duomenys.
------------------------	---

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

```
template<typename T >
ManoVector & ManoVector< T >::operator= (
    ManoVector< T > && Kitas_Vektorius ) [inline]
```

Move operatorius.

##### Parameters

<i>Kitas_Vektorius</i>	Vektorius, iš kurio perkeliama duomenys.
------------------------	--

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

```
template<typename T >
T & ManoVector< T >::operator[] (
    unsigned int indeksas ) [inline]
```

Operatorius, kuris grąžina nurodytą elementą.

**Parameters**

<i>indeksas</i>	Elemento indeksas.
-----------------	--------------------

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

```
template<typename T >
const T & ManoVector< T >::operator[] (
    unsigned int indeksas ) const [inline]
```

Operatorius, kuris grąžina nurodytą elementą.

**Parameters**

<i>indeksas</i>	Elemento indeksas.
-----------------	--------------------

**4.1.3.26 pop\_back()**

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

Funkcija, kuri pašalina elementą iš **ManoVector** pabaigos.

**4.1.3.27 push\_back()**

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

Funkcija, kuri prideda elementą į vektoriaus pabaigą.

**Parameters**

<i>value</i>	Elementas, kuris turi būti pridėtas.
--------------	--------------------------------------

**4.1.3.28 reserve()**

```
template<typename T >
```



```
void ManoVector< T >::reserve (
    unsigned int nauja_talpa ) [inline]
```

Funkcija, kuri rezervuoja vietas vektoriuje.

#### Parameters

<i>nauja_talpa</i>	Nauja vektoriaus talpa.
--------------------	-------------------------

#### Exceptions

<i>std::length_error</i>	Jei vektorius neturi tiek vietos.
--------------------------	-----------------------------------

#### 4.1.3.29 resize()

```
template<typename T >
void ManoVector< T >::resize (
    unsigned int naujas_dydis ) [inline]
```

Funkcija, kuri pakeičia `ManoVector` dydį į nurodytą dydį.

#### Parameters

<i>naujas_dydis</i>	Naujas <code>ManoVector</code> dydis.
---------------------	---------------------------------------

#### 4.1.3.30 shrink\_to\_fit()

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

Funkcija, kuri sumažina vektoriaus talpą iki vektoriaus dydžio.

#### 4.1.3.31 size()

```
template<typename T >
unsigned int ManoVector< T >::size ( ) const [inline]
```

Funkcija, kuri grąžina vektoriaus dydį.

#### Returns

Vektoriaus dydis.

#### 4.1.3.32 swap()

```
template<typename T >
void ManoVector< T >::swap (
    ManoVector< T > & Kitas_Vektorius ) [inline]
```

Funkcija, kuri sukeičia du vektorius vietomis.

## Parameters

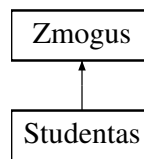
<i>Kitas_Vektorius</i>	Kitas vektorius.
------------------------	------------------

## 4.2 Studentas Class Reference

Studento klasė.

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

Inheritance diagram for Studentas:



### Public Member Functions

- [Studentas](#) ()  
*Default'inis Studento konstruktorius.*
- [Studentas](#) (const string &vard, const string &pavard)  
*Namų darbų gavimui.*
- [ManoVector](#)< int > [Get\\_Nd](#) () const  
*Namų darbų gavimui.*
- int [Get\\_Egzaminas](#) () const  
*Gauti egzamino rezultata.*
- double [Get\\_Mediana](#) () const  
*Gauti galutinį balą pagal medianą.*
- double [Get\\_Vidurkis](#) () const  
*Gauti galutinį balą pagal vidurkį.*
- bool [Nd\\_empty](#) () const  
*Patikrinti ar namų darbų vektorius yra tuščias.*
- int [Nd\\_dydis](#) () const  
*Namų darbų vektoriaus dydis.*
- void [nd\\_rusiavimas](#) ()  
*Namų darbų rūšiavimas didėjimo tvarka.*
- int [Nd\\_Suma](#) ()  
*Apskaičiuoti namų darbų sumą.*
- int [Get\\_Last\\_Nd](#) ()  
*Gauti namų darbų vektoriaus paskutinį pažymį.*
- [~Studentas](#) ()  
*Studento klasės destruktorius.*
- void [setEgzaminas](#) (int egz)  
*Priskirti egzamino rezultata.*
- void [SetMediana](#) (double med)  
*Priskirti medianos balą.*

- void [setVidurkis](#) (double vid)  
*Priskirti vidurkio balą.*
- void [setNd](#) (int nd)  
*Pridėti namų darbo rezultatą i vektorių.*
- void [DeleteLastNd](#) ()  
*Ištrinti paskutinį namų darbų rezultatą.*
- void [ClearEverything](#) ()  
*Išvalyti visą studentą.*
- void [ND\\_clear](#) ()  
*Išvalyti namų darbų vektorių.*
- double [Vidurkis](#) (int nd\_kiekis, int nd\_suma, int egzaminas)  
*Apskaičiuoti galutinį balą pagal vidurkį.*
- double [medianoSkaiciavimas](#) (const [ManoVector](#)< int > &namu\_darbai, int nd\_kiekis, int egzaminas)  
*Apskaičiuoti galutinį balą pagal medianą.*
- void [Print](#) () const override  
*Tik tam, kad būtų galima paveldėti [Zmogus](#) klasę.*
- [Studentas](#) (const [Studentas](#) &LaikinasStudentas)  
*Kopijavimo konstruktorius.*
- [Studentas](#) ([Studentas](#) &&LaikinasStudentas) noexcept  
*Perkėlimo konstruktorius.*
- [Studentas](#) & [operator=](#) (const [Studentas](#) &LaikinasStudentas)  
*Kopijavimo priskyrimo operatorius.*
- [Studentas](#) & [operator=](#) ([Studentas](#) &&LaikinasStudentas)  
*Perkėlimo priskyrimo operatorius.*

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

- string [Get\\_Vardas](#) () const  
*Vardo gavimui.*
- string [Get\\_Pavarde](#) () const  
*Pavardės gavimui.*
- void [SetVardas](#) (string vard)  
*Vardo nustatymui.*
- void [SetPavarde](#) (string pav)  
*Pavardės nustatymui.*

## Friends

- istream & [operator>>](#) (istream &filename, [Studentas](#) &LaikinasStudentas)  
*Perkrautas įvesties operatorius darbui su failais.*
- istream & [operator>>](#) (istream &>manual, [Studentas](#) &LaikinasStudentas)  
*Perkrautas įvesties operatorius darbui su vartotoju per konsolę.*
- ostream & [operator<<](#) (ostream &console, const [Studentas](#) &LaikinasStudentas)  
*Perkrautas išvesties operatorius į konsolę.*
- ofstream & [operator<<](#) (ofstream &filename, const [Studentas](#) &LaikinasStudentas)  
*Perkrautas išvesties operatorius į failą.*

## Additional Inherited Members

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

- [Zmogus](#) ()
- [Zmogus](#) (const string &vard, const string &pavard)  
*Default'inis konstruktorius.*
- virtual [~Zmogus](#) ()  
*[Zmogus](#) klasės destruktorius.*

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

- string [vardas](#)
- string [pavarde](#)  
*Saugo vardą ir pavardę*

## 4.2.1 Detailed Description

Studento klasė.

## 4.2.2 Constructor & Destructor Documentation

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

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

Default'inis Studento konstruktorius.

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

```
Studentas::Studentas (
    const string & vard,
    const string & pavard )
```

Namų darbų gavimui.

#### Returns

Namų darbų vektorių.

### 4.2.2.3 ~Studentas()

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

Studento klasės destruktorius.

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

```
Studentas::Studentas (
    const Studentas & LaikinasStudentas )
```

Kopijavimo konstruktorius.

## Parameters

<i>LaikinasStudentas</i>	Objektas, kurį reikia kopijuoti.
--------------------------	----------------------------------

**4.2.2.5 Studentas()** [4/4]

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

Perkėlimo konstruktorius.

## Parameters

<i>LaikinasStudentas</i>	Objektas, kurį reikia perkelti.
--------------------------	---------------------------------

**4.2.3 Member Function Documentation****4.2.3.1 ClearEverything()**

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

Išvalyti visą studentą.

**4.2.3.2 DeleteLastNd()**

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

Ištrinti paskutinį namų darbų rezultatą.

**4.2.3.3 Get\_Egzaminas()**

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

Gauti egzamino rezultatą.

## Returns

Egzamino rezultatą.

**4.2.3.4 Get\_Last\_Nd()**

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

Gauti namų darbų vektoriaus paskutinį pažymį.

## Returns

Paskutinį pažymį namų darbų vektoriuje.

#### 4.2.3.5 Get\_Mediana()

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

Gauti galutinį balą pagal medianą.

##### Returns

Galutinis balas pagal medianą.

#### 4.2.3.6 Get\_Nd()

```
ManoVector< int > Studentas::Get_Nd ( ) const [inline]
```

Namų darbų gavimui.

##### Returns

Namų darbų vektorius.

#### 4.2.3.7 Get\_Vidurkis()

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

Gauti galutinį balą pagal vidurkį.

##### Returns

galutinis balas pagal vidurkį.

#### 4.2.3.8 medianosSkaiciavimas()

```
double Studentas::medianosSkaiciavimas (
    const ManoVector< int > & namu_darbai,
    int nd_kiekis,
    int egzaminas )
```

Apskaičiuoti galutinį balą pagal medianą.

##### Parameters

<i>namu_darbai</i>	Namų darbų vektorius.
<i>nd_kiekis</i>	Namų darbų rezultatų kiekis.
<i>egzaminas</i>	Egzamino rezultatas.

**Returns**

Galutinį balą pagal medianą.

**4.2.3.9 ND\_clear()**

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

Išvalyti namų darbų vektorių.

**4.2.3.10 Nd\_dydis()**

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

Namų darbų vektoriaus dydis.

**Returns**

Namų darbų vektoriaus dydį.

**4.2.3.11 Nd\_empty()**

```
bool Studentas::Nd_empty ( ) const [inline]
```

Patikrinti ar namų darbų vektorius yra tuščias.

**Returns**

True, jei vektorius tuščias. False, jei vektorius turi elementų.

**4.2.3.12 nd\_rusiavimas()**

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

Namų darbų rūšiavimas didėjimo tvarka.

**4.2.3.13 Nd\_Suma()**

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

Apskaičiuoti namų darbų sumą.

**Returns**

namų darbų vektoriaus elementų sumą.

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

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

Kopijavimo priskyrimo operatorius.

## Parameters

<i>LaikinasStudentas</i>	Objektas, kurį reikia kopijuoti.
--------------------------	----------------------------------

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

```
Studentas & Studentas::operator= (
    Studentas && LaikinasStudentas )
```

Perkėlimo priskyrimo operatorius.

## Parameters

<i>LaikinasStudentas</i>	Objektas, kurį reikia perkelti.
--------------------------	---------------------------------

**4.2.3.16 Print()**

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

Tik tam, kad būtų galima paveldėti [Zmogus](#) klasę.

Implements [Zmogus](#).

**4.2.3.17 setEgzaminas()**

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

Priskirti egzamino rezultatą.

## Parameters

<i>egz</i>	egzamino rezultatas.
------------	----------------------

**4.2.3.18 SetMediana()**

```
void Studentas::SetMediana (
    double med ) [inline]
```

Priskirti medianos balą.

## Parameters

<i>med</i>	medianos balas.
------------	-----------------



**4.2.3.19 setNd()**

```
void Studentas::setNd (
    int nd ) [inline]
```

Pridėti namų darbo rezultatą į vektorių.

**Parameters**

<i>nd</i>	Namų darbo rezultatas.
-----------	------------------------

**4.2.3.20 setVidurkis()**

```
void Studentas::setVidurkis (
    double vid ) [inline]
```

Priskirti vidurkio balą.

**Parameters**

<i>vid</i>	vidurkio balas.
------------	-----------------

**4.2.3.21 Vidurkis()**

```
double Studentas::Vidurkis (
    int nd_kiekis,
    int nd_suma,
    int egzaminas )
```

Apskaičiuoti galutinį balą pagal vidurkį.

**Parameters**

<i>nd_kiekis</i>	Namų darbų rezultatų kiekis.
<i>nd_suma</i>	Visų namų darbų suma.
<i>egzaminas</i>	Egzamino rezultatas.

**Returns**

Galutinį balą pagal vidurkį.

**4.2.4 Friends And Related Symbol Documentation****4.2.4.1 operator<< [1/2]**

```
ofstream & operator<< (
    ofstream & filename,
    const Studentas & LaikinasStudentas ) [friend]
```

Perkrautas išvesties operatorius į failą.

**Parameters**

<i>filename</i>	Išvesties ofstream objektas.
<i>LaikinasStudentas</i>	Objektas, kurį reikia išvesti.

**Returns**

Išvesties ofstream objektą.

**4.2.4.2 operator<< [2/2]**

```
ostream & operator<< (
    ostream & console,
    const Studentas & LaikinasStudentas ) [friend]
```

Perkrautas išvesties operatorius į konsolę.

**Parameters**

<i>console</i>	Išvesties ostream objektas.
<i>LaikinasStudentas</i>	Objektas, kurį reikia išvesti.

**Returns**

Išvesties ostream objektą.

**4.2.4.3 operator>> [1/2]**

```
istream & operator>> (
    istream & manual,
    Studentas & LaikinasStudentas ) [friend]
```

Perkrautas įvesties operatorius darbui su vartotoju per konsolę.

**Parameters**

<i>manual</i>	Įvesties istream objektas.
<i>LaikinasStudentas</i>	Objektas į kurį reikia skaityti duomenis.

**Returns**

Įvesties istream objektą.

**4.2.4.4 operator>> [2/2]**

```
istreamstream & operator>> (
    istreamstream & filename,
    Studentas & LaikinasStudentas ) [friend]
```

Perkrautas įvesties operatorius darbui su failais.

#### Parameters

<i>filename</i>	Įvesties stringstream objektas.
<i>LaikinasStudentas</i>	Objektas į kurį reikia skaityti duomenis.

#### Returns

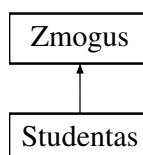
Įvesties stringstream objektą.

## 4.3 Zmogus Class Reference

Abstrakti žmogaus klasė.

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

Inheritance diagram for Zmogus:



#### Public Member Functions

- string [Get\\_Vardas](#) () const  
*Vardo gavimui.*
- string [Get\\_Pavarde](#) () const  
*Pavardės gavimui.*
- void [SetVardas](#) (string vard)  
*Vardo nustatymui.*
- void [SetPavarde](#) (string pav)  
*Pavardės nustatymui.*
- virtual void [Print](#) () const =0  
*Spausdinimo funkcija padaranti klasę abstrakčia.*

#### Protected Member Functions

- [Zmogus](#) ()
- [Zmogus](#) (const string &vard, const string &pavard)  
*Default'inis konstruktorius.*
- virtual [~Zmogus](#) ()  
*[Zmogus](#) klasės destruktorius.*

## Protected Attributes

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

*Saugo vardą ir pavardę*

### 4.3.1 Detailed Description

Abstrakti žmogaus klasė.

[Zmogus](#) klasė yra abstrakti klasė, kuri turi tik vardą ir pavardę. Ji turi virtualią funkciją [Print\(\)](#), kuri yra privaloma klasėms, kurios paveldi [Zmogus](#) klasę.

### 4.3.2 Constructor & Destructor Documentation

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

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

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

```
Zmogus::Zmogus (
    const string & vard,
    const string & pavard ) [inline], [protected]
```

Default'inis konstruktorius.

[Zmogus](#) klasės konstruktorius.

#### Parameters

<i>vard</i>	Studento vardas.
<i>pavard</i>	Studento pavardė.

#### 4.3.2.3 ~Zmogus()

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

[Zmogus](#) klasės destruktorius.

### 4.3.3 Member Function Documentation

#### 4.3.3.1 Get\_Pavarde()

```
string Zmogus::Get_Pavarde ( ) const [inline]
```

Pavardės gavimui.

**Returns**

Pavarde.

**4.3.3.2 Get\_Vardas()**

```
string Zmogus::Get_Vardas ( ) const [inline]
```

Vardo gavimui.

**Returns**

Vardą.

**4.3.3.3 Print()**

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

Spausdinimo funkcija padaranti klasę abstrakčia.

Implemented in [Studentas](#).

**4.3.3.4 SetPavarde()**

```
void Zmogus::SetPavarde (
    string pav ) [inline]
```

Pavardės nustatymui.

**Parameters**

<i>pav</i>	Pavardė.
------------	----------

**4.3.3.5 SetVardas()**

```
void Zmogus::SetVardas (
    string vard ) [inline]
```

Vardo nustatymui.

**Parameters**

<i>vard</i>	Vardas.
-------------	---------

### 4.3.4 Member Data Documentation

#### 4.3.4.1 pavarde

```
string Zmogus::pavarde [protected]
```

Saugo vardą ir pavardę

#### 4.3.4.2 vardas

```
string Zmogus::vardas [protected]
```

## Chapter 5

# File Documentation

### 5.1 Headers/funkcijos.h File Reference

```
#include "Zmogus.h"
#include "Vector.h"
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <limits>
#include <numeric>
#include <random>
#include <ctime>
#include <fstream>
#include <chrono>
```

#### Classes

- class [Studentas](#)  
*Studento klasė.*

#### Functions

- bool [palygintiPagalVarda](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Palyginti du studentus pagal vardą.*
- bool [palygintiPagalPavarde](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Palyginti su studentus pagal pavardę.*
- bool [palygintiPagalVidurki](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Palyginti dviejų studentų balus pagal vidurkį.*
- bool [palygintiPagalMediana](#) (const [Studentas](#) &a, const [Studentas](#) &b)  
*Palyginti dviejų studentų balus pagal medianą.*
- void [PasalintiKietusStudentus](#) ([ManoVector](#)< [Studentas](#) > &[Studentai](#), int [norimas\\_rikiavimas](#))  
*Panaikinti studentus, kurių galutinis balas  $\geq 5.0$ .*
- void [GeneruotiFaila](#) (int kiekis, int nd\_kiekis)  
*Generuoti studentų failą.*
- void [GeneruotiPazymius](#) ([Studentas](#) &[LaikinasStudentas](#))  
*Generuoti atsitiktinius pažymius studentui.*
- void [GeneruotiStudenta](#) ([Studentas](#) &[LaikinasStudentas](#))  
*Generuoti atsitiktinį studentą.*
- void [IsvestiRezultatus](#) (string pavadinimas, const [ManoVector](#)< [Studentas](#) > &[Studentai](#), int [norima\\_isvedimo\\_vieta](#))  
*Išvesti rezultatus į ekraną arba į failą.*

## Variables

- int `norima_isvedimo_vieta`  
*Kintamasis, saugantis vartotojo norimą rezultatų išvedimo vietą*
- int `programos_veikimas`  
*Kintamasis, saugantis vartotojo norimą programos veikimą*
- int `norimas_rikiavimas`  
*Kintamasis, saugantis vartotojo norimą rikiavimą*
- char `programos_tesinys`  
*Kintamasis, kuris žino ar vartotojas nori testuoti darbą su programa.*
- char `choice3`  
*Kintamasis, kuris žino ar vartotojas nori kartoti tam tikrą programos dalį (Pvz.: įvesti namų darbo rezultatą)*
- `ManoVector< string > Vardai`  
*Vardų vektorius skirtas generuoti atsitiktinius vardus.*
- `ManoVector< string > Pavardes`  
*Pavardžių vektorius skirtas generuoti atsitiktines pavardes.*
- `ManoVector< Studentas > Lievi`  
*Studentų vektorius, kurie gavo skolą*
- `ManoVector< Studentas > Studentai`  
*Studentų vektorius, kurie išlaikė dalyką*

## 5.1.1 Function Documentation

### 5.1.1.1 GeneruotiFaila()

```
void GeneruotiFaila (
    int kiekis,
    int nd_kiekis )
```

Generuoti studentų failą.

#### Parameters

<code>kiekis</code>	Norimas studentų kiekis.
<code>nd_kiekis</code>	Norimas namų darbų kiekis kiekvienam studentui.

### 5.1.1.2 GeneruotiPazymius()

```
void GeneruotiPazymius (
    Studentas & LaikinasStudentas )
```

Generuoti atsitiktinius pažymius studentui.

#### Parameters

<code>LaikinasStudentas</code>	Studento objektas.
--------------------------------	--------------------



### 5.1.1.3 GeneruotiStudenta()

```
void GeneruotiStudenta (
    Studentas & LaikinasStudentas )
```

Generuoti atsitiktinį studentą.

#### Parameters

<i>LaikinasStudentas</i>	Studento objektas.
--------------------------	--------------------

### 5.1.1.4 IsvestiRezultatus()

```
void IsvestiRezultatus (
    string pavadinimas,
    const ManoVector< Studentas > & Studentai,
    int norima_isvedimo_vieta )
```

Išvesti rezultatus į ekraną arba į failą.

#### Parameters

<i>pavadinimas</i>	Norimas failo pavadinimas.
<i>Studentai</i>	Studentų vektorius.
<i>norima_isvedimo_vieta</i>	Norima išvedimo vieta.

### 5.1.1.5 palygintiPagalMediana()

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

Palyginti dviejų studentų balus pagal medianą.

#### Parameters

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

#### Returns

True, jei a studento galutinis balas pagal medianą yra mažesnis nei b studento, false atvirkščiai.

### 5.1.1.6 palygintiPagalPavarde()

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

Palyginti su studentus pagal pavarde.

## Parameters

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

## Returns

True, jei a studentu pavardė yra žemiau pagal abėcėlę nei b studento, false atvirkščiai.

**5.1.1.7 palygintiPagalVarda()**

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

Palyginti du studentus pagal vardą.

## Parameters

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

## Returns

True, jei a studento vardas yra žemiau pagal abėcėlę nei b studento, false atvirkščiai.

**5.1.1.8 palygintiPagalVidurki()**

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

Palyginti dviejų studentų balus pagal vidurkį.

## Parameters

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

## Returns

True, jei a studento galutinis balas pagal vidurkį yra mažesnis nei b studento, false atvirkščiai.

**5.1.1.9 PasalintiKietusStudentus()**

```
void PasalintiKietusStudentus (
    ManoVector< Studentas > & Studentai,
    int norimas_rikiavimas )
```

Panaikinti studentus, kuriu galutinis balas  $\geq 5.0$ .

#### Parameters

<i>Studentai</i>	Studentų vektorius.
<i>norimas_rikiavimas</i>	Norimas rikiavimas.

## 5.1.2 Variable Documentation

### 5.1.2.1 choice3

```
char choice3 [extern]
```

Kintamasis, kuris žino ar vartotojas nori kartoti tam tikrą programos dalį (Pvz.: įvesti namų darbo rezultatą)

### 5.1.2.2 Lievi

```
ManoVector<Studentas> Lievi [extern]
```

Studentų vektorius, kurie gavo skolą

### 5.1.2.3 norima\_isvedimo\_vieta

```
int norima_isvedimo_vieta [extern]
```

Kintamasis, saugantis vartotojo norimą rezultatų išvedimo vietą

### 5.1.2.4 norimas\_rikiavimas

```
int norimas_rikiavimas [extern]
```

Kintamasis, saugantis vartotojo norimą rikiavimą

### 5.1.2.5 Pavardes

```
ManoVector<string> Pavardes [extern]
```

Pavardžių vektorius skirtas generuoti atsitiktines pavardes.

### 5.1.2.6 programos\_tesinys

```
char programos_tesinys [extern]
```

Kintamasis, kuris žino ar vartotojas nori testuoti darbą su programa.

### 5.1.2.7 programos\_veikimas

```
int programos_veikimas [extern]
```

Kintamasis, saugantis vartotojo norimą programos veikimą

### 5.1.2.8 Studentai

```
ManoVector<Studentas> Studentai [extern]
```

Studentų vektorius, kurie išlaikė dalyką

### 5.1.2.9 Vardai

```
ManoVector<string> Vardai [extern]
```

Vardų vektorius skirtas generuoti atsitiktinius vardus.

## 5.2 funkcijos.h

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

```
00001 #ifndef FUNKCIJOS_H
00002 #define FUNKCIJOS_H
00003
00004 #include "Zmogus.h"
00005 #include "Vector.h"
00006 #include <iostream>
00007 #include <iomanip>
00008 #include <algorithm>
00009 #include <limits>
00010 #include <numeric>
00011 #include <random>
00012 #include <ctime>
00013 #include <fstream>
00014 #include <chrono>
00015
00016 using namespace std;
00017
00021 class Studentas : public Zmogus {
00022     private:
00023         ManoVector<int> namu_darbai;
00024         int egzaminas;
00025         double mediana;
00026         double vidurkis;
00027     public:
00031         Studentas();
00032
00037         Studentas(const string &vard, const string &pavard);
00038
00043         ManoVector<int> Get_Nd() const { return namu_darbai; }
00044
00049         int Get_Egzaminas() const { return egzaminas; }
00050
00055         double Get_Mediana() const { return mediana; }
00056
00061         double Get_Vidurkis() const { return vidurkis; }
00062
00067         bool Nd_empty() const { return namu_darbai.empty(); }
00068
00073         int Nd_dydis() const { return namu_darbai.size(); }
00074
00078         void nd_rusivimas() { sort(namu_darbai.begin(), namu_darbai.end()); }
00079
00084         int Nd_Suma() { return accumulate(namu_darbai.begin(), namu_darbai.end(), 0); }
00085
00090         int Get_Last_Nd() { return namu_darbai.back(); }
00091
```

```

00095     ~Studentas();
00096
00101     void setEgzaminas(int egz) { this->egzaminas = egz; }
00102
00107     void SetMediana(double med) { this->mediana = med; }
00108
00113     void setVidurkis(double vid) { this->vidurkis = vid; }
00114
00119     void setNd(int nd) { this->namu_darbai.push_back(move(nd)); }
00120
00124     void DeleteLastNd() { this->namu_darbai.pop_back(); }
00125
00129     void ClearEverything() { this->vardas.clear(); this->pavarde.clear();
this->namu_darbai.clear(); this->egzaminas = 0; this->mediana = 0.0; this->vidurkis = 0.0; }
00130
00134     void ND_clear() { this->namu_darbai.clear(); }
00135
00143     double Vidurkis(int nd_kiekis, int nd_suma, int egzaminas);
00144
00152     double medianosSkaiciavimas(const ManoVector<int> &namu_darbai, int nd_kiekis, int egzaminas);
00153
00157     void Print() const override {};
00158
00163     Studentas(const Studentas &LaikinasStudentas);
00164
00169     Studentas(Studentas &&LaikinasStudentas) noexcept;
00170
00175     Studentas& operator=(const Studentas &LaikinasStudentas);
00176
00181     Studentas& operator=(Studentas &&LaikinasStudentas);
00182
00189     friend istream& operator>(istream& filename, Studentas &LaikinasStudentas);
00190
00197     friend istream& operator>(istream& manual, Studentas &LaikinasStudentas);
00198
00205     friend ostream& operator<(ostream& console, const Studentas &LaikinasStudentas);
00206
00213     friend ofstream& operator<<(ofstream& filename, const Studentas &LaikinasStudentas);
00214 };
00215
00216 extern int norima_isvedimo_vieta;
00217 extern int programos_veikimas;
00218 extern int norimas_rikiavimas;
00219 extern char programos_tesinys;
00220 extern char choice3;
00222 extern ManoVector<string> Vardai;
00223 extern ManoVector<string> Pavardes;
00224 extern ManoVector<Studentas> Lievi;
00225 extern ManoVector<Studentas> Studentai;
00233 bool palygintiPagalVarda(const Studentas &a, const Studentas &b);
00234
00241 bool palygintiPagalPavarde(const Studentas &a, const Studentas &b);
00242
00249 bool palygintiPagalVidurki(const Studentas &a, const Studentas &b);
00250
00257 bool palygintiPagalMediana(const Studentas &a, const Studentas &b);
00258
00264 void PasalintiKietusStudentus(ManoVector<Studentas> &Studentai, int norimas_rikiavimas);
00265
00271 void GeneruotiFaila(int kiekis, int nd_kiekis);
00272
00277 void GeneruotiPazymius(Studentas &LaikinasStudentas);
00278
00283 void GeneruotiStudenta(Studentas &LaikinasStudentas);
00284
00291 void IvestiRezultatus(string pavadinimas, const ManoVector<Studentas> &Studentai, int
norima_isvedimo_vieta);
00292
00293 #endif

```

## 5.3 Headers/Vector.h File Reference

```

#include <iostream>
#include <stdexcept>
#include <limits>
#include <algorithm>
#include <initializer_list>
#include <string>

```

## Classes

- class `ManoVector< T >`

*Vektoriaus klasė pritaikyta šitai programai.*

## 5.4 Vector.h

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

```

00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <iostream>
00005 #include <stdexcept>
00006 #include <limits>
00007 #include <algorithm>
00008 #include <initializer_list>
00009 #include <string>
00010
00016 template <typename T>
00017 class ManoVector {
00018     private:
00019         size_t dydis;
00020         size_t talpa;
00021         T* duomenys;
00022     public:
00023         //Member funkcijos
00024
00029         int max_size() const { return std::numeric_limits<unsigned int>::max() / sizeof(T); }
00030         ManoVector() : dydis(0), talpa(0), duomenys(new T[talpa]) {}
00031         ManoVector(std::initializer_list<T> il) : dydis(il.size()), talpa(il.size()), duomenys(new
T[talpa]) {
00032             std::copy(il.begin(), il.end(), duomenys);
00033         }
00039         ManoVector(const ManoVector<T>& Kitas_Vektorius) : dydis(Kitas_Vektorius.dydis),
talpa(Kitas_Vektorius.talpa), duomenys(new T[talpa]) {
00040             for (unsigned int i = 0; i < dydis; i++) {
00041                 duomenys[i] = Kitas_Vektorius.duomenys[i];
00042             }
00043         }
00044
00049         ManoVector(ManoVector&& Kitas_Vektorius) noexcept : dydis(Kitas_Vektorius.dydis),
talpa(Kitas_Vektorius.talpa), duomenys(Kitas_Vektorius.duomenys) {
00050             Kitas_Vektorius.dydis = 0;
00051             Kitas_Vektorius.talpa = 0;
00052             Kitas_Vektorius.duomenys = nullptr;
00053         }
00054
00055         ~ManoVector() { delete[] duomenys; }
00061         ManoVector& operator=(const ManoVector& Kitas_Vektorius) {
00062             if (this == &Kitas_Vektorius) return *this;
00063             delete[] duomenys;
00064             dydis = Kitas_Vektorius.dydis;
00065             talpa = Kitas_Vektorius.talpa;
00066             duomenys = new T[talpa];
00067             for (unsigned int i = 0; i < dydis; i++) {
00068                 duomenys[i] = Kitas_Vektorius.duomenys[i];
00069             }
00070             return *this;
00071         }
00072
00077         ManoVector& operator=(ManoVector&& Kitas_Vektorius) {
00078             if (this == &Kitas_Vektorius) return *this;
00079             delete[] duomenys;
00080             dydis = Kitas_Vektorius.dydis;
00081             talpa = Kitas_Vektorius.talpa;
00082             duomenys = Kitas_Vektorius.duomenys;
00083             Kitas_Vektorius.dydis = 0;
00084             Kitas_Vektorius.talpa = 0;
00085             Kitas_Vektorius.duomenys = nullptr;
00086             return *this;
00087         }
00088
00089         //Element access funkcijos
00090
00095         T& operator[](unsigned int indeksas) {return duomenys[indeksas];}
00096
00101         const T& operator[](unsigned int indeksas) const {return duomenys[indeksas];}
00102
00109         T& at(unsigned int indeksas) {

```

```

00110         if (indeksas >= dydis) {
00111             throw std::out_of_range("Indeksas už ribų");
00112         }
00113         return duomenys[indeksas];
00114     }
00115
00122     const T& at(unsigned int indeksas) const {
00123         if (indeksas >= dydis) {
00124             throw std::out_of_range("Indeksas už ribų");
00125         }
00126         return duomenys[indeksas];
00127     }
00128
00133     T& front() {return duomenys[0];}
00134
00139     const T& front() const {return duomenys[0];}
00140
00145     T& back() {return duomenys[dydis - 1];}
00146
00151     const T& back() const {return duomenys[dydis - 1];}
00152
00157     T* data() noexcept {return duomenys;}
00158
00159     //Iterators funkcijos
00160
00165     T* begin() noexcept {return duomenys;}
00166
00171     const T* begin() const noexcept {return duomenys;}
00172
00177     T* end() noexcept {return duomenys + dydis;}
00178
00183     const T* end() const noexcept {return duomenys + dydis;}
00184
00185     //Capacity funkcijos
00186
00191     unsigned int capacity() const {return talpa;}
00192
00197     unsigned int size() const {return dydis;}
00198
00203     bool empty() const {return dydis == 0;}
00204
00211     void reserve(unsigned int nauja_talpa) {
00212         if (nauja_talpa <= talpa) return;
00213         if (nauja_talpa > max_size()) throw std::length_error("Vektorius tiek vietos neturi:");
00214         T* nauji_duomenys = new T[nauja_talpa];
00215         for (unsigned int i = 0; i < dydis; i++) {
00216             nauji_duomenys[i] = duomenys[i];
00217         }
00218         delete[] duomenys;
00219         duomenys = nauji_duomenys;
00220         talpa = nauja_talpa;
00221     }
00222
00226     void shrink_to_fit() {
00227         if (dydis == talpa) return;
00228         T* nauji_duomenys = new T[dydis];
00229         for (unsigned int i = 0; i < dydis; i++) {
00230             nauji_duomenys[i] = duomenys[i];
00231         }
00232         delete[] duomenys;
00233         duomenys = nauji_duomenys;
00234         talpa = dydis;
00235     }
00236
00242     void assign(unsigned int n, const T& value) {
00243         if (n > talpa) reserve(n);
00244         for (unsigned int i = 0; i < n; i++) {
00245             duomenys[i] = value;
00246         }
00247         dydis = n;
00248     }
00249
00250     //Modifiers funkcijos
00251     void clear() {dydis = 0;}
00257     void push_back(const T& value) {
00258         if (dydis >= talpa) reserve(talpa == 0 ? 1 : dydis * 2);
00259         duomenys[dydis++] = value;
00260     }
00261
00266     T* insert(unsigned int indeksas, const T& value) {
00267         if (dydis >= talpa) reserve(talpa == 0 ? 1 : talpa * 2);
00268         for (unsigned int i = dydis; i > indeksas; i--) {
00269             duomenys[i] = duomenys[i - 1];
00270         }
00271         duomenys[indeksas] = value;
00272         dydis++;
00273         return &duomenys[indeksas];

```



```

00274     }
00275
00280     T* emplace(unsigned int indeksas, T&& value) {
00281         if (dydis == talpa) reserve(talpa == 0 ? 1 : talpa * 2);
00282         for (unsigned int i = dydis; i > indeksas; i--) {
00283             duomenys[i] = duomenys[i - 1];
00284         }
00285         duomenys[indeksas] = std::move(value);
00286         dydis++;
00287         return &duomenys[indeksas];
00288     }
00289
00294     T& emplace_back(T&& value) {
00295         if (dydis >= talpa) reserve(talpa == 0 ? 1 : dydis * 2);
00296         duomenys[dydis++] = std::move(value);
00297         return duomenys[dydis - 1];
00298     }
00299
00303     void pop_back() {
00304         if (dydis > 0) {
00305             --dydis;
00306         }
00307     }
00308
00313     void resize(unsigned int naujas_dydis) {
00314         if (naujas_dydis > talpa) reserve(naujas_dydis);
00315         for (unsigned int i = dydis; i < naujas_dydis; i++) {
00316             duomenys[i] = T();
00317         }
00318         dydis = naujas_dydis;
00319     }
00320
00325     void swap(ManoVector& Kitas_Vektorius) {
00326         std::swap(dydis, Kitas_Vektorius.dydis);
00327         std::swap(talpa, Kitas_Vektorius.talpa);
00328         std::swap(duomenys, Kitas_Vektorius.duomenys);
00329     }
00330
00335     T* erase(unsigned int indeksas) {
00336         for (unsigned int i = indeksas; i < dydis - 1; i++) {
00337             duomenys[i] = duomenys[i + 1];
00338         }
00339         dydis--;
00340         return &duomenys[indeksas];
00341     }
00342
00349     T* erase(T* pirmas_elementas, T* paskutinis_elementas) {
00350         if (pirmas_elementas >= duomenys && paskutinis_elementas <= duomenys + dydis) {
00351             size_t ElementaiSalinimui = paskutinis_elementas - pirmas_elementas;
00352             size_t ElementaiPerkelimui = duomenys + dydis - paskutinis_elementas;
00353
00354             for (size_t i = 0; i < ElementaiPerkelimui; i++) {
00355                 *(pirmas_elementas + i) = *(paskutinis_elementas + i);
00356             }
00357
00358             dydis -= ElementaiSalinimui;
00359
00360             if (paskutinis_elementas == duomenys + dydis) {
00361                 return paskutinis_elementas;
00362             }
00363         }
00364         return paskutinis_elementas;
00365     }
00366 };
00367
00368 #endif

```

## 5.5 Headers/Zmogus.h File Reference

```
#include <string>
```

### Classes

- class [Zmogus](#)

*Abstrakti žmogaus klasė.*

## 5.6 Zmogus.h

Go to the documentation of this file.

```
00001 #ifndef ZMOGUS_H
00002 #define ZMOGUS_H
00003
00004 #include <string>
00005 using namespace std;
00006
00015 class Zmogus {
00016     protected:
00017         string vardas, pavarde;
00019         Zmogus() : vardas(""), pavarde("") {};
00025         Zmogus(const string &vard, const string &pavard) : vardas(vard), pavarde(pavard) {};
00029         virtual ~Zmogus() {};
00030     public:
00035         inline string Get_Vardas() const { return vardas; }
00040         inline string Get_Pavarde() const { return pavarde; }
00045         void SetVardas(string vard) { this->vardas = vard; }
00050         void SetPavarde(string pav) { this->pavarde = pav; }
00054         virtual void Print() const = 0;
00055 };
00056
00057 #endif
```

## 5.7 Sources/funkcijos.cpp File Reference

```
#include "../Headers/Zmogus.h"
#include "../Headers/funkcijos.h"
```

### Functions

- `istreamstream & operator>>` (`istreamstream &filename`, `Studentas &LaikinasStudentas`)
- `istream & operator>>` (`istream &>manual`, `Studentas &LaikinasStudentas`)
- `ostream & operator<<` (`ostream &console`, `const Studentas &LaikinasStudentas`)
- `ofstream & operator<<` (`ofstream &filename`, `const Studentas &LaikinasStudentas`)
- `bool palygintiPagalVarda` (`const Studentas &a`, `const Studentas &b`)  
*Palyginti du studentus pagal vardą.*
- `bool palygintiPagalPavarde` (`const Studentas &a`, `const Studentas &b`)  
*Palyginti su studentus pagal pavarde.*
- `bool palygintiPagalVidurki` (`const Studentas &a`, `const Studentas &b`)  
*Palyginti dviejų studentų balus pagal vidurkį.*
- `bool palygintiPagalMediana` (`const Studentas &a`, `const Studentas &b`)  
*Palyginti dviejų studentų balus pagal medianą.*
- `void PasalintiKietusStudentus` (`ManoVector< Studentas > &Studentai`, `int norimas_rikiavimas`)  
*Panaikinti studentus, kuriu galutinis balas >= 5.0.*
- `void GeneruotiFaila` (`int kiekis`, `int nd_kiekis`)  
*Generuoti studentų failą.*
- `void IvestiRezultatus` (`string pavadinimas`, `const ManoVector< Studentas > &Studentai`, `int norima_isvedimo_vieta`)  
*Išvesti rezultatus į ekraną arba į failą.*
- `void GeneruotiPazymius` (`Studentas &LaikinasStudentas`)  
*Generuoti atsitiktinius pažymius studentui.*
- `void GeneruotiStudenta` (`Studentas &LaikinasStudentas`)  
*Generuoti atsitiktinį studentą.*

## 5.7.1 Function Documentation

### 5.7.1.1 GeneruotiFaila()

```
void GeneruotiFaila (
    int kiekis,
    int nd_kiekis )
```

Generuoti studentų failą.

#### Parameters

<i>kiekis</i>	Norimas studentų kiekis.
<i>nd_kiekis</i>	Norimas namų darbų kiekis kiekvienam studentui.

### 5.7.1.2 GeneruotiPazymius()

```
void GeneruotiPazymius (
    Studentas & LaikinasStudentas )
```

Generuoti atsitiktinius pažymius studentui.

#### Parameters

<i>LaikinasStudentas</i>	Studento objektas.
--------------------------	--------------------

### 5.7.1.3 GeneruotiStudenta()

```
void GeneruotiStudenta (
    Studentas & LaikinasStudentas )
```

Generuoti atsitiktinį studentą.

#### Parameters

<i>LaikinasStudentas</i>	Studento objektas.
--------------------------	--------------------

### 5.7.1.4 IsvestiRezultatus()

```
void IsvestiRezultatus (
    string pavadinimas,
    const ManoVector< Studentas > & Studentai,
    int norima_isvedimo_vieta )
```

Išvesti rezultatus į ekraną arba į failą.

## Parameters

<i>pavadinimas</i>	Norimas failo pavadinimas.
<i>Studentai</i>	Studentų vektorius.
<i>norima_isvedimo_vieta</i>	Norima išvedimo vieta.

**5.7.1.5 operator<<()** [1/2]

```
ofstream & operator<< (
    ofstream & filename,
    const Studentas & LaikinasStudentas )
```

## Parameters

<i>filename</i>	Išvesties ofstream objektas.
<i>LaikinasStudentas</i>	Objektas, kurį reikia išvesti.

## Returns

Išvesties ofstream objektą.

**5.7.1.6 operator<<()** [2/2]

```
ostream & operator<< (
    ostream & console,
    const Studentas & LaikinasStudentas )
```

## Parameters

<i>console</i>	Išvesties ostream objektas.
<i>LaikinasStudentas</i>	Objektas, kurį reikia išvesti.

## Returns

Išvesties ostream objektą.

**5.7.1.7 operator>>()** [1/2]

```
istream & operator>> (
    istream & manual,
    Studentas & LaikinasStudentas )
```

## Parameters

<i>manual</i>	Išvesties istream objektas.
<i>LaikinasStudentas</i>	Objektas į kurį reikia skaityti duomenis.

**Returns**

Įvesties istream objektą.

**5.7.1.8 operator>>() [2/2]**

```
istream & operator>> (
    istream & filename,
    Studentas & LaikinasStudentas )
```

**Parameters**

<i>filename</i>	Įvesties stringstream objektas.
<i>LaikinasStudentas</i>	Objektas į kurį reikia skaityti duomenis.

**Returns**

Įvesties stringstream objektą.

**5.7.1.9 palygintiPagalMediana()**

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

Palyginti dviejų studentų balus pagal medianą.

**Parameters**

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

**Returns**

True, jei a studento galutinis balas pagal medianą yra mažesnis nei b studento, false atvirkščiai.

**5.7.1.10 palygintiPagalPavarde()**

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

Palyginti su studentus pagal pavardę.

**Parameters**

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

**Returns**

True, jei a studentu pavardė yra žemiau pagal abėcėlę nei b studento, false atvirkščiai.

**5.7.1.11 palygintiPagalVarda()**

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

Palyginti du studentus pagal vardą.

**Parameters**

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

**Returns**

True, jei a studento vardas yra žemiau pagal abėcėlę nei b studento, false atvirkščiai.

**5.7.1.12 palygintiPagalVidurki()**

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

Palyginti dviejų studentų balus pagal vidurkį.

**Parameters**

<i>a</i>	Pirmasis studentas.
<i>b</i>	Antrasis studentas.

**Returns**

True, jei a studento galutinis balas pagal vidurkį yra mažesnis nei b studento, false atvirkščiai.

**5.7.1.13 PasalintiKietusStudentus()**

```
void PasalintiKietusStudentus (
    ManoVector< Studentas > & Studentai,
    int norimas_rikiavimas )
```

Panaikinti studentus, kuriu galutinis balas  $\geq 5.0$ .

## Parameters

<i>Studentai</i>	Studentų vektorius.
<i>norimas_rikiavimas</i>	Norimas rikiavimas.

## 5.8 Sources/Vector\_v2\_0.cpp File Reference

```
#include "../Headers/Zmogus.h"
#include "../Headers/funkcijos.h"
#include <vector>
```

## Functions

- int [main](#) ()

## Variables

- const int [N](#) = 10
- int [programos\\_veikimas](#)  
*Kintamasis, saugantis vartotojo norimą programos veikimą*

### 5.8.1 Function Documentation

#### 5.8.1.1 main()

```
int main ( )
```

### 5.8.2 Variable Documentation

#### 5.8.2.1 N

```
const int N = 10
```

#### 5.8.2.2 [programos\\_veikimas](#)

```
int programos\_veikimas
```

*Kintamasis, saugantis vartotojo norimą programos veikimą*

## 5.9 Testavimas/Testavimas.cpp File Reference

```
#include "../Headers/funkcijos.h"
#include <gtest/gtest.h>
```

### Functions

- [TEST](#) (Studento\_Testavimas, Studento\_Default\_Konstruktorius)
- [TEST](#) (Studento\_Testavimas, Studento\_Parametirinis\_Konstruktorius)
- [TEST](#) (Studento\_Testavimas, Studento\_Move\_konstruktorius)
- [TEST](#) (Studento\_Testavimas, Studento\_Copy\_konstruktorius)
- [TEST](#) (Studento\_Testavimas, Studento\_kopijavimo\_operatorius)
- [TEST](#) (Studento\_Testavimas, Studento\_move\_operatorius)
- [TEST](#) (Studento\_Testavimas, Studento\_Seteriai\_ir\_Geteriai)
- int [main](#) (int argc, char \*\*argv)

### Variables

- int [programos\\_veikimas](#)  
*Kintamasis, saugantis vartotojo norimą programos veikimą*

### 5.9.1 Function Documentation

#### 5.9.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

#### 5.9.1.2 TEST() [1/7]

```
TEST (
    Studento_Testavimas ,
    Studento_Copy_konstruktorius )
```

#### 5.9.1.3 TEST() [2/7]

```
TEST (
    Studento_Testavimas ,
    Studento_Default_Konstruktorius )
```

#### 5.9.1.4 TEST() [3/7]

```
TEST (
    Studento_Testavimas ,
    Studento_kopijavimo_operatorius )
```



#### 5.9.1.5 TEST() [4/7]

```
TEST (
    Studento_Testavimas ,
    Studento_Move_konstruktorius )
```

#### 5.9.1.6 TEST() [5/7]

```
TEST (
    Studento_Testavimas ,
    Studento_move_operatorius )
```

#### 5.9.1.7 TEST() [6/7]

```
TEST (
    Studento_Testavimas ,
    Studento_Parametirinis_Konstruktorius )
```

#### 5.9.1.8 TEST() [7/7]

```
TEST (
    Studento_Testavimas ,
    Studento_Seteriai_ir_Geteriai )
```

### 5.9.2 Variable Documentation

#### 5.9.2.1 programos\_veikimas

```
int programos_veikimas
```

Kintamasis, saugantis vartotojo norimą programos veikimą

## 5.10 Testavimas/Vektoriaus\_Testavimas.cpp File Reference

```
#include "../Headers/Vector.h"
#include <gtest/gtest.h>
```

## Functions

- [TEST](#) (Vektoriaus\_Member\_Funkciju\_Testavimas, konstruktorius)
- [TEST](#) (Vektoriaus\_Member\_Funkciju\_Testavimas, Kopijavimo\_Konstruktorius)
- [TEST](#) (Vektoriaus\_Member\_Funkciju\_Testavimas, Move\_Konstruktorius)
- [TEST](#) (Vektoriaus\_Member\_Funkciju\_Testavimas, Kopijavimo\_Operatorius)
- [TEST](#) (Vektoriaus\_Member\_Funkciju\_Testavimas, initializer\_list)
- [TEST](#) (Vektoriaus\_Member\_Funkciju\_Testavimas, Move\_Operatorius)
- [TEST](#) (Vektoriaus\_Element\_Access\_Funkciju\_Testavimas, Operatorius\_Kvadratinu\_Sklaustu)
- [TEST](#) (Vektoriaus\_Element\_Access\_Funkciju\_Testavimas, At\_Funkcija)
- [TEST](#) (Vektoriaus\_Element\_Access\_Funkciju\_Testavimas, Front\_Funkcija)
- [TEST](#) (Vektoriaus\_Element\_Access\_Funkciju\_Testavimas, Back\_Funkcija)
- [TEST](#) (Vektoriaus\_Element\_Access\_Funkciju\_Testavimas, Data\_Funkcija)
- [TEST](#) (Vektoriaus\_Iterator\_Funkciju\_Testavimas, Begin\_Funkcija)
- [TEST](#) (Vektoriaus\_Iterator\_Funkciju\_Testavimas, End\_Funkcija)
- [TEST](#) (Vektoriaus\_Capacity\_Funkciju\_Testavimas, Capacity\_Funkcija)
- [TEST](#) (Vektoriaus\_Capacity\_Funkciju\_Testavimas, Size\_Funkcija)
- [TEST](#) (Vektoriaus\_Capacity\_Funkciju\_Testavimas, Empty\_Funkcija)
- [TEST](#) (Vektoriaus\_Capacity\_Funkciju\_Testavimas, Reserve\_Funkcija)
- [TEST](#) (Vektoriaus\_Capacity\_Funkciju\_Testavimas, Shrink\_to\_fit\_Funkcija)
- [TEST](#) (Vektoriaus\_Capacity\_Funkciju\_Testavimas, Assign\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Clear\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Push\_Back\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Emplace\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Emplace\_Back\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Pop\_Back\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Resize\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Swap\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Insert\_Funkcija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Erase\_Funkcija\_Pagal\_Pozicija)
- [TEST](#) (Vektoriaus\_Modifiers\_Funkciju\_Testavimas, Erase\_Funkcija)
- [int main](#) (int argc, char \*\*argv)

## 5.10.1 Function Documentation

### 5.10.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

### 5.10.1.2 TEST() [1/29]

```
TEST (
    Vektoriaus_Capacity_Funkciju_Testavimas ,
    Assign_Funkcija )
```

### 5.10.1.3 TEST() [2/29]

```
TEST (
    Vektoriaus_Capacity_Funkciju_Testavimas ,
    Capacity_Funkcija )
```

**5.10.1.4 TEST() [3/29]**

```
TEST (
    Vektoriaus_Capacity_Funkciju_Testavimas ,
    Empty_Funkcija )
```

**5.10.1.5 TEST() [4/29]**

```
TEST (
    Vektoriaus_Capacity_Funkciju_Testavimas ,
    Reserve_Funkcija )
```

**5.10.1.6 TEST() [5/29]**

```
TEST (
    Vektoriaus_Capacity_Funkciju_Testavimas ,
    Shrink_to_fit_Funkcija )
```

**5.10.1.7 TEST() [6/29]**

```
TEST (
    Vektoriaus_Capacity_Funkciju_Testavimas ,
    Size_Funkcija )
```

**5.10.1.8 TEST() [7/29]**

```
TEST (
    Vektoriaus_Element_Access_Funkciju_Testavimas ,
    At_Funkcija )
```

**5.10.1.9 TEST() [8/29]**

```
TEST (
    Vektoriaus_Element_Access_Funkciju_Testavimas ,
    Back_Funkcija )
```

**5.10.1.10 TEST() [9/29]**

```
TEST (
    Vektoriaus_Element_Access_Funkciju_Testavimas ,
    Data_Funkcija )
```

**5.10.1.11 TEST() [10/29]**

```
TEST (
    Vektoriaus_Element_Access_Funkciju_Testavimas ,
    Front_Funkcija )
```

**5.10.1.12 TEST() [11/29]**

```
TEST (
    Vektoriaus_Element_Access_Funkciju_Testavimas ,
    Operatorius_Kvadratinu_Sklaustu )
```

**5.10.1.13 TEST() [12/29]**

```
TEST (
    Vektoriaus_Iterator_Funkciju_Testavimas ,
    Begin_Funkcija )
```

**5.10.1.14 TEST() [13/29]**

```
TEST (
    Vektoriaus_Iterator_Funkciju_Testavimas ,
    End_Funkcija )
```

**5.10.1.15 TEST() [14/29]**

```
TEST (
    Vektoriaus_Member_Funkciju_Testavimas ,
    initializer_list )
```

**5.10.1.16 TEST() [15/29]**

```
TEST (
    Vektoriaus_Member_Funkciju_Testavimas ,
    konstruktorius )
```

**5.10.1.17 TEST() [16/29]**

```
TEST (
    Vektoriaus_Member_Funkciju_Testavimas ,
    Kopijavimo_Konstruktorius )
```

**5.10.1.18 TEST() [17/29]**

```
TEST (
    Vektoriaus_Member_Funkciju_Testavimas ,
    Kopijavimo_Operatorius )
```

**5.10.1.19 TEST() [18/29]**

```
TEST (
    Vektoriaus_Member_Funkciju_Testavimas ,
    Move_Konstruktorius )
```

**5.10.1.20 TEST()** [19/29]

```
TEST (
    Vektoriaus_Member_Funkciju_Testavimas ,
    Move_Operatorius )
```

**5.10.1.21 TEST()** [20/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Clear_Funkcija )
```

**5.10.1.22 TEST()** [21/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Eplace_Back_Funkcija )
```

**5.10.1.23 TEST()** [22/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Eplace_Funkcija )
```

**5.10.1.24 TEST()** [23/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Erase_Funkcija )
```

**5.10.1.25 TEST()** [24/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Erase_Funkcija_Pagal_Pozicija )
```

**5.10.1.26 TEST()** [25/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Insert_Funkcija )
```

**5.10.1.27 TEST()** [26/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Pop_Back_Funkcija )
```

**5.10.1.28 TEST()** [27/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Push_Back_Funkcija )
```

**5.10.1.29 TEST()** [28/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Resize_Funkcija )
```

**5.10.1.30 TEST()** [29/29]

```
TEST (
    Vektoriaus_Modifiers_Funkciju_Testavimas ,
    Swap_Funkcija )
```

# Index

- ~ManoVector
  - ManoVector< T >, [10](#)
- ~Studentas
  - Studentas, [20](#)
- ~Zmogus
  - Zmogus, [28](#)
- assign
  - ManoVector< T >, [10](#)
- at
  - ManoVector< T >, [10](#)
- back
  - ManoVector< T >, [11](#)
- begin
  - ManoVector< T >, [11](#)
- capacity
  - ManoVector< T >, [12](#)
- choice3
  - funkcijos.h, [36](#)
- clear
  - ManoVector< T >, [12](#)
- ClearEverything
  - Studentas, [21](#)
- data
  - ManoVector< T >, [12](#)
- DeleteLastNd
  - Studentas, [21](#)
- emplace
  - ManoVector< T >, [12](#)
- emplace\_back
  - ManoVector< T >, [13](#)
- empty
  - ManoVector< T >, [13](#)
- end
  - ManoVector< T >, [13](#)
- erase
  - ManoVector< T >, [13](#), [14](#)
- front
  - ManoVector< T >, [14](#)
- funkcijos.cpp
  - GeneruotiFaila, [43](#)
  - GeneruotiPazymius, [43](#)
  - GeneruotiStudenta, [43](#)
  - IsvestiRezultatus, [43](#)
  - operator<<, [44](#)
  - operator>>, [44](#), [45](#)
  - palygintiPagalMediana, [45](#)
  - palygintiPagalPavarde, [45](#)
  - palygintiPagalVarda, [46](#)
  - palygintiPagalVidurki, [46](#)
  - PasalintiKietusStudentus, [46](#)
- funkcijos.h
  - choice3, [36](#)
  - GeneruotiFaila, [32](#)
  - GeneruotiPazymius, [32](#)
  - GeneruotiStudenta, [32](#)
  - IsvestiRezultatus, [33](#)
  - Lievi, [36](#)
  - norima\_isvedimo\_vieta, [36](#)
  - norimas\_rikiavimas, [36](#)
  - palygintiPagalMediana, [33](#)
  - palygintiPagalPavarde, [33](#)
  - palygintiPagalVarda, [35](#)
  - palygintiPagalVidurki, [35](#)
  - PasalintiKietusStudentus, [35](#)
  - Pavardes, [36](#)
  - programos\_tesinys, [36](#)
  - programos\_veikimas, [36](#)
  - Studentai, [37](#)
  - Vardai, [37](#)
- GeneruotiFaila
  - funkcijos.cpp, [43](#)
  - funkcijos.h, [32](#)
- GeneruotiPazymius
  - funkcijos.cpp, [43](#)
  - funkcijos.h, [32](#)
- GeneruotiStudenta
  - funkcijos.cpp, [43](#)
  - funkcijos.h, [32](#)
- Get\_Egzaminas
  - Studentas, [21](#)
- Get\_Last\_Nd
  - Studentas, [21](#)
- Get\_Mediana
  - Studentas, [21](#)
- Get\_Nd
  - Studentas, [22](#)
- Get\_Pavarde
  - Zmogus, [28](#)
- Get\_Vardas
  - Zmogus, [29](#)
- Get\_Vidurkis
  - Studentas, [22](#)
- Headers/funkcijos.h, [31](#), [37](#)

Headers/Vector.h, [38](#), [39](#)  
 Headers/Zmogus.h, [41](#), [42](#)  
 insert  
     ManoVector< T >, [14](#)  
 IsvestiRezultatus  
     funkcijos.cpp, [43](#)  
     funkcijos.h, [33](#)  
 Lievi  
     funkcijos.h, [36](#)  
 main  
     Testavimas.cpp, [48](#)  
     Vector\_v2\_0.cpp, [47](#)  
     Vektoriaus\_Testavimas.cpp, [50](#)  
 ManoVector  
     ManoVector< T >, [9](#)  
 ManoVector< T >, [7](#)  
     ~ManoVector, [10](#)  
     assign, [10](#)  
     at, [10](#)  
     back, [11](#)  
     begin, [11](#)  
     capacity, [12](#)  
     clear, [12](#)  
     data, [12](#)  
     emplace, [12](#)  
     emplace\_back, [13](#)  
     empty, [13](#)  
     end, [13](#)  
     erase, [13](#), [14](#)  
     front, [14](#)  
     insert, [14](#)  
     ManoVector, [9](#)  
     max\_size, [15](#)  
     operator=, [15](#)  
     operator[], [16](#)  
     pop\_back, [16](#)  
     push\_back, [16](#)  
     reserve, [16](#)  
     resize, [17](#)  
     shrink\_to\_fit, [17](#)  
     size, [17](#)  
     swap, [17](#)  
 max\_size  
     ManoVector< T >, [15](#)  
 medianosSkaiciavimas  
     Studentas, [22](#)  
 N  
     Vector\_v2\_0.cpp, [47](#)  
 ND\_clear  
     Studentas, [23](#)  
 Nd\_dydis  
     Studentas, [23](#)  
 Nd\_empty  
     Studentas, [23](#)  
 nd\_rusiavimas  
     Studentas, [23](#)  
 Nd\_Suma  
     Studentas, [23](#)  
 norima\_isvedimo\_vieta  
     funkcijos.h, [36](#)  
 norimas\_rikiavimas  
     funkcijos.h, [36](#)  
 operator<<  
     funkcijos.cpp, [44](#)  
     Studentas, [25](#), [26](#)  
 operator>>  
     funkcijos.cpp, [44](#), [45](#)  
     Studentas, [26](#)  
 operator=  
     ManoVector< T >, [15](#)  
     Studentas, [23](#), [24](#)  
 operator[]  
     ManoVector< T >, [16](#)  
 palygintiPagalMediana  
     funkcijos.cpp, [45](#)  
     funkcijos.h, [33](#)  
 palygintiPagalPavarde  
     funkcijos.cpp, [45](#)  
     funkcijos.h, [33](#)  
 palygintiPagalVarda  
     funkcijos.cpp, [46](#)  
     funkcijos.h, [35](#)  
 palygintiPagalVidurki  
     funkcijos.cpp, [46](#)  
     funkcijos.h, [35](#)  
 PasalintiKietusStudentus  
     funkcijos.cpp, [46](#)  
     funkcijos.h, [35](#)  
 pavarde  
     Zmogus, [30](#)  
 Pavardes  
     funkcijos.h, [36](#)  
 pop\_back  
     ManoVector< T >, [16](#)  
 Print  
     Studentas, [24](#)  
     Zmogus, [29](#)  
 programos\_tesinys  
     funkcijos.h, [36](#)  
 programos\_veikimas  
     funkcijos.h, [36](#)  
     Testavimas.cpp, [49](#)  
     Vector\_v2\_0.cpp, [47](#)  
 push\_back  
     ManoVector< T >, [16](#)  
 reserve  
     ManoVector< T >, [16](#)  
 resize  
     ManoVector< T >, [17](#)  
 setEgzaminas



- Studentas, [24](#)
- SetMediana
  - Studentas, [24](#)
- setNd
  - Studentas, [24](#)
- SetPavarde
  - Zmogus, [29](#)
- SetVardas
  - Zmogus, [29](#)
- setVidurkis
  - Studentas, [25](#)
- shrink\_to\_fit
  - ManoVector< T >, [17](#)
- size
  - ManoVector< T >, [17](#)
- Sources/funkcijos.cpp, [42](#)
- Sources/Vector\_v2\_0.cpp, [47](#)
- Studentai
  - funkcijos.h, [37](#)
- Studentas, [18](#)
  - ~Studentas, [20](#)
  - ClearEverything, [21](#)
  - DeleteLastNd, [21](#)
  - Get\_Egzaminas, [21](#)
  - Get\_Last\_Nd, [21](#)
  - Get\_Mediana, [21](#)
  - Get\_Nd, [22](#)
  - Get\_Vidurkis, [22](#)
  - medianosSkaiciavimas, [22](#)
  - ND\_clear, [23](#)
  - Nd\_dydis, [23](#)
  - Nd\_empty, [23](#)
  - nd\_rusiavimas, [23](#)
  - Nd\_Suma, [23](#)
  - operator<<, [25](#), [26](#)
  - operator>>, [26](#)
  - operator=, [23](#), [24](#)
  - Print, [24](#)
  - setEgzaminas, [24](#)
  - SetMediana, [24](#)
  - setNd, [24](#)
  - setVidurkis, [25](#)
  - Studentas, [20](#), [21](#)
  - Vidurkis, [25](#)
- swap
  - ManoVector< T >, [17](#)
- TEST
  - Testavimas.cpp, [48](#), [49](#)
  - Vektoriaus\_Testavimas.cpp, [50–54](#)
- Testavimas.cpp
  - main, [48](#)
  - programos\_veikimas, [49](#)
  - TEST, [48](#), [49](#)
- Testavimas/Testavimas.cpp, [48](#)
- Testavimas/Vektoriaus\_Testavimas.cpp, [49](#)
- Vardai
  - funkcijos.h, [37](#)
- vardas
  - Zmogus, [30](#)
- Vector\_v2\_0.cpp
  - main, [47](#)
  - N, [47](#)
  - programos\_veikimas, [47](#)
- Vektoriaus\_Testavimas.cpp
  - main, [50](#)
  - TEST, [50–54](#)
- Vidurkis
  - Studentas, [25](#)
- Zmogus, [27](#)
  - ~Zmogus, [28](#)
  - Get\_Pavarde, [28](#)
  - Get\_Vardas, [29](#)
  - pavarde, [30](#)
  - Print, [29](#)
  - SetPavarde, [29](#)
  - SetVardas, [29](#)
  - vardas, [30](#)
  - Zmogus, [28](#)