

My Project V3.0

Generated by Doxygen 1.10.0

1 V3 Versija	1
1.1 Kas nauja 3 versijoje	1
1.2 Funkciju testavimas	1
1.2.1 Vektorių palyginimai	1
1.2.2 Clear, pop, erase	1
1.2.3 empty, getsize, get capacity	1
1.2.4 first, last, begin, end	1
1.2.5 [], at, pop	1
1.3 myVector ir std::vector spartos testavimas	1
1.4 Resize Kiekis	2
1.4.1 Atminties persikirstymas myVector konteineryje yra 1 mažesnis dėl to, kad default myVector sukuriamas su 1 capacity	2
1.5 Programos veikimo laikas su konteineriais	2
1.5.1 100.000.txt	2
1.5.2 1.000.000.txt	2
1.5.3 10.000.000.txt	2
1.5.4 Papildomas testavimas	2
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 myVector< T > Class Template Reference	9
5.1.1 Constructor & Destructor Documentation	10
5.1.1.1 myVector() [1/6]	10
5.1.1.2 myVector() [2/6]	10
5.1.1.3 myVector() [3/6]	10
5.1.1.4 myVector() [4/6]	10
5.1.1.5 myVector() [5/6]	10
5.1.1.6 myVector() [6/6]	10
5.1.1.7 ~myVector()	11
5.1.2 Member Function Documentation	11
5.1.2.1 assign() [1/2]	11
5.1.2.2 assign() [2/2]	11
5.1.2.3 at()	11
5.1.2.4 begin()	11
5.1.2.5 clear()	11
5.1.2.6 empty()	11

5.1.2.7 end()	12
5.1.2.8 erase() [1/2]	12
5.1.2.9 erase() [2/2]	12
5.1.2.10 first()	12
5.1.2.11 getCapacity()	12
5.1.2.12 insert() [1/3]	12
5.1.2.13 insert() [2/3]	12
5.1.2.14 insert() [3/3]	13
5.1.2.15 last()	13
5.1.2.16 operator!=(())	13
5.1.2.17 operator<()	13
5.1.2.18 operator<=()	13
5.1.2.19 operator=() [1/2]	13
5.1.2.20 operator=() [2/2]	13
5.1.2.21 operator==(())	14
5.1.2.22 operator>()	14
5.1.2.23 operator>=()	14
5.1.2.24 operator[]()	14
5.1.2.25 pop_back()	14
5.1.2.26 print()	14
5.1.2.27 push_back()	14
5.1.2.28 reserve()	14
5.1.2.29 resize()	15
5.1.2.30 size()	15
5.1.2.31 swap()	15
5.2 studentai_base Class Reference	15
5.2.1 Constructor & Destructor Documentation	16
5.2.1.1 studentai_base()	16
5.2.1.2 ~studentai_base()	16
5.2.2 Member Function Documentation	16
5.2.2.1 setMasyvas()	16
5.2.2.2 testav()	16
5.2.3 Member Data Documentation	16
5.2.3.1 balai_	16
5.2.3.2 dydis_	16
5.2.3.3 egzaminas_	16
5.2.3.4 mediana_	16
5.2.3.5 pavarde_	17
5.2.3.6 rodykle_	17
5.2.3.7 vardas_	17
5.2.3.8 vidurkis_	17
5.3 studentai_class Class Reference	17

5.3.1 Constructor & Destructor Documentation	18
5.3.1.1 studentai_class() [1/5]	18
5.3.1.2 studentai_class() [2/5]	18
5.3.1.3 ~studentai_class()	18
5.3.1.4 studentai_class() [3/5]	18
5.3.1.5 studentai_class() [4/5]	19
5.3.1.6 studentai_class() [5/5]	19
5.3.2 Member Function Documentation	19
5.3.2.1 getBalai()	19
5.3.2.2 getEgzaminas()	19
5.3.2.3 getMediana()	19
5.3.2.4 getPavarde()	19
5.3.2.5 getVardas()	19
5.3.2.6 getVidurkis()	19
5.3.2.7 operator=() [1/2]	19
5.3.2.8 operator=() [2/2]	20
5.3.2.9 setBalai()	20
5.3.2.10 setEgzaminas()	20
5.3.2.11 setMediana()	20
5.3.2.12 setPavarde()	20
5.3.2.13 setVardas()	20
5.3.2.14 setVidurkis()	20
5.3.2.15 testav()	20
5.3.3 Friends And Related Symbol Documentation	21
5.3.3.1 operator<<	21
5.3.3.2 operator>>	21
6 File Documentation	23
6.1 bendrosFunkcijos.cpp File Reference	23
6.1.1 Function Documentation	23
6.1.1.1 distribution()	23
6.1.1.2 extraSpace()	23
6.1.1.3 failoGeneracija()	24
6.1.1.4 laikoSausdinimas()	24
6.1.1.5 mt()	24
6.1.1.6 tarpai()	24
6.1.2 Variable Documentation	24
6.1.2.1 rd	24
6.2 funkcijosVektoriai.cpp File Reference	24
6.2.1 Function Documentation	25
6.2.1.1 antrasPasirinkimas()	25
6.2.1.2 darbasSuVektoriais()	25

6.2.1.3	klasiuTestavimas()	25
6.2.1.4	Less()	25
6.2.1.5	LessM()	25
6.2.1.6	medianosApsk()	26
6.2.1.7	NuskaitymasFailo()	26
6.2.1.8	pirmasPasirinkimas()	26
6.2.1.9	rusiavimasMediana()	26
6.2.1.10	rusiavimasPavarde()	26
6.2.1.11	rusiavimasVardas()	26
6.2.1.12	rusiavimasVidurkis()	26
6.2.1.13	rusiavimoMenu()	26
6.2.1.14	rusiavimoMenuSkirstymas()	27
6.2.1.15	skirstymas()	27
6.2.1.16	spausdinimasFaile()	27
6.2.1.17	spausdinimasFaileSkirstymas()	27
6.2.1.18	spausdinimasTerminale()	27
6.2.1.19	spausdinimasTerminaleSkirstymas()	27
6.2.1.20	treciasPasirinkimas()	27
6.2.1.21	vektoriuTestavimas()	27
6.2.1.22	vidurkioApsk()	27
6.3	FunkcijuBaze.h File Reference	27
6.3.1	Function Documentation	28
6.3.1.1	darbasSuDekais()	28
6.3.1.2	darbasSuListais()	28
6.3.1.3	darbasSuVektoriais()	28
6.3.1.4	extraSpace()	28
6.3.1.5	failoGeneracija()	28
6.3.1.6	tarpai()	28
6.4	FunkcijuBaze.h	29
6.5	funkcijuBazeVektoriai.h File Reference	29
6.5.1	Function Documentation	30
6.5.1.1	antrasPasirinkimas()	30
6.5.1.2	extraSpace()	30
6.5.1.3	failoGeneracija()	30
6.5.1.4	klasiuTestavimas()	30
6.5.1.5	laikoSpuausdinimas()	30
6.5.1.6	medianosApsk()	30
6.5.1.7	NuskaitymasFailo()	30
6.5.1.8	pirmasPasirinkimas()	30
6.5.1.9	rusiavimasMediana()	31
6.5.1.10	rusiavimasPavarde()	31
6.5.1.11	rusiavimasVardas()	31

6.5.1.12 rusiavimasVidurkis()	31
6.5.1.13 rusiavimoMenu()	31
6.5.1.14 rusiavimoMenuSkirstymas()	31
6.5.1.15 skirstymas()	31
6.5.1.16 spausdinimasFaile()	31
6.5.1.17 spausdinimasFaileSkirstymas()	31
6.5.1.18 spausdinimasTerminale()	32
6.5.1.19 spausdinimasTerminaleSkirstymas()	32
6.5.1.20 tarpai()	32
6.5.1.21 treciasPasirinkimas()	32
6.5.1.22 vektoriuTestavimas()	32
6.5.1.23 vidurkioApsk()	32
6.6 funkcijuBazeVektoriai.h	32
6.7 Includes.h File Reference	33
6.7.1 Variable Documentation	34
6.7.1.1 distribution	34
6.7.1.2 mt	34
6.7.1.3 rd	34
6.8 Includes.h	34
6.9 klasesRealizacija.cpp File Reference	34
6.9.1 Function Documentation	35
6.9.1.1 operator<<()	35
6.9.1.2 operator>>()	35
6.10 main.cpp File Reference	35
6.10.1 Function Documentation	35
6.10.1.1 main()	35
6.11 myVector.h File Reference	35
6.12 myVector.h	36
6.13 readMe.md File Reference	39

Index	41
--------------	-----------

Chapter 1

V3 Versija

1.1 Kas nauja 3 versjoje

1. Sukurta nuosava vektoriaus klase - [myVector](#)
2. Patikrintos funkcijos
3. Atlikti testavimai

1.2 Funkciju testavimas

1.2.1 Vektorių palyginimai

1.2.2 Clear, pop, erase

1.2.3 empty, getsize, get capacity

1.2.4 first, last, begin, end

1.2.5 [], at, pop

1.3 myVector ir std::vector spartos testavimas

Elementų kiekis	std::vector	myVector
10000	0.0001257	0.0001122
100000	0.0008885	0.000937
1000000	0.0077361	0.0071594
10000000	0.078231	0.0986152
100000000	0.725096	0.785338

1. [myVector](#) veikia greičiau su mažesniais skaičiais (Vidutiniškai 10% - 15% greičiau)

2. Su dideliais skaičiais `myVector` veikia lėčiau (Vidutiniškai 5% - 10% lėčiau)

1.4 Resize Kiekis

- 1.4.1 Atminties perskirstymas `myVector` konteineryje yra 1 mažesnins dėl to, kad default `myVector` sukuriamas su 1 capacity

1.5 Programos veikimo laikas su konteineriais

1.5.1 100.000.txt

Darbas	<code>std::vector</code>	<code>myVector</code>
Duomenų nuskaitymas	0.2695445	0.2869592
skirstymas į dvi grupes	0.0452824	0.1115644
Rusiavimas	0.2883375	0.3545991
Visas laikas	0.6031644	0.7531227

1.5.2 1.000.000.txt

Darbas	<code>std::vector</code>	<code>myVector</code>
Duomenų nuskaitymas	2.609202	2.738449
skirstymas į dvi grupes	0.5331666	1.227727
Rusiavimas	3.764685	4.563468
Visas laikas	6.907054	8.529645

1.5.3 10.000.000.txt

Darbas	<code>std::vector</code>	<code>myVector</code>
Duomenų nuskaitymas	27.49252	30.6472
skirstymas į dvi grupes	4.522078	11.05496
Rusiavimas	46.87415	58.14309
Visas laikas	78.88875	99.84526

1.5.4 Papildomas testavimas

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

myVector< T >	9
myVector< int >	9
studentai_base	15
studentai_class	17

Chapter 3

Class Index

3.1 Class List

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

myVector< T >	9
studentai_base	15
studentai_class	17

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

bendrosFunkcijos.cpp	23
funkcijosVektoriai.cpp	24
FunkcijuBaze.h	27
funkcijuBazeVektoriai.h	29
Includes.h	33
klasesRealizacija.cpp	34
main.cpp	35
myVector.h	35

Chapter 5

Class Documentation

5.1 myVector< T > Class Template Reference

```
#include <myVector.h>
```

Public Member Functions

- [myVector \(\)](#)
- [myVector \(size_t value, T data\)](#)
- [myVector \(std::initializer_list< T > ilist\)](#)
- [myVector \(T data\)](#)
- [myVector \(const myVector &other\)](#)
- [myVector \(myVector &&other\) noexcept](#)
- [myVector & operator= \(const myVector &other\)](#)
- [myVector & operator= \(myVector &&other\) noexcept](#)
- [~myVector \(\)](#)
- [void reserve \(size_t new_capacity\)](#)
- [void push_back \(T duomenys\)](#)
- [void pop_back \(\)](#)
- [T at \(size_t index\)](#)
- [int size \(\) const](#)
- [int getCapacity \(\)](#)
- [void print \(\)](#)
- [bool empty \(\)](#)
- [void clear \(\)](#)
- [T & first \(\)](#)
- [T & last \(\)](#)
- [T * begin \(\)](#)
- [T * end \(\)](#)
- [T & operator\[\] \(size_t index\) const](#)
- [void assign \(size_t count, const T &value\)](#)
- [template<typename InputIterator > void assign \(InputIterator first, InputIterator last\)](#)
- [void erase \(T *position\)](#)
- [void erase \(T *first, T *last\)](#)
- [void swap \(myVector &other\)](#)
- [void resize \(size_t n\)](#)

- `T * insert (T *position, const T &val)`
- `void insert (T *position, size_t n, const T &val)`
- `template<typename InputIterator >`
`void insert (T *position, InputIterator first, InputIterator last)`
- `bool operator== (const myVector &other) const`
- `bool operator!= (const myVector &other) const`
- `bool operator< (const myVector &other) const`
- `bool operator<= (const myVector &other) const`
- `bool operator> (const myVector &other) const`
- `bool operator>= (const myVector &other) const`

5.1.1 Constructor & Destructor Documentation

5.1.1.1 myVector() [1/6]

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

5.1.1.2 myVector() [2/6]

```
template<typename T >
myVector< T >::myVector (
    size_t value,
    T data ) [inline]
```

5.1.1.3 myVector() [3/6]

```
template<typename T >
myVector< T >::myVector (
    std::initializer_list< T > ilist ) [inline]
```

5.1.1.4 myVector() [4/6]

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

5.1.1.5 myVector() [5/6]

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

5.1.1.6 myVector() [6/6]

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

5.1.1.7 ~myVector()

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

5.1.2 Member Function Documentation

5.1.2.1 assign() [1/2]

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

5.1.2.2 assign() [2/2]

```
template<typename T >
void myVector< T >::assign (
    size_t count,
    const T & value ) [inline]
```

5.1.2.3 at()

```
template<typename T >
T myVector< T >::at (
    size_t index ) [inline]
```

5.1.2.4 begin()

```
template<typename T >
T * myVector< T >::begin ( ) [inline]
```

5.1.2.5 clear()

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

5.1.2.6 empty()

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

5.1.2.7 end()

```
template<typename T >
T * myVector< T >::end ( ) [inline]
```

5.1.2.8 erase() [1/2]

```
template<typename T >
void myVector< T >::erase (
    T * first,
    T * last ) [inline]
```

5.1.2.9 erase() [2/2]

```
template<typename T >
void myVector< T >::erase (
    T * position ) [inline]
```

5.1.2.10 first()

```
template<typename T >
T & myVector< T >::first ( ) [inline]
```

5.1.2.11 getCapacity()

```
template<typename T >
int myVector< T >::getCapacity ( ) [inline]
```

5.1.2.12 insert() [1/3]

```
template<typename T >
T * myVector< T >::insert (
    T * position,
    const T & val ) [inline]
```

5.1.2.13 insert() [2/3]

```
template<typename T >
template<typename InputIterator >
void myVector< T >::insert (
    T * position,
    InputIterator first,
    InputIterator last ) [inline]
```

5.1.2.14 insert() [3/3]

```
template<typename T >
void myVector< T >::insert (
    T * position,
    size_t n,
    const T & val ) [inline]
```

5.1.2.15 last()

```
template<typename T >
T & myVector< T >::last ( ) [inline]
```

5.1.2.16 operator"!="()

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

5.1.2.17 operator<()

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

5.1.2.18 operator<=()

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

5.1.2.19 operator=() [1/2]

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

5.1.2.20 operator=() [2/2]

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

5.1.2.21 operator==()

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

5.1.2.22 operator>()

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

5.1.2.23 operator>=()

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

5.1.2.24 operator[]()

```
template<typename T >
T & myVector< T >::operator[] (
    size_t index ) const [inline]
```

5.1.2.25 pop_back()

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

5.1.2.26 print()

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

5.1.2.27 push_back()

```
template<typename T >
void myVector< T >::push_back (
    T duomenys ) [inline]
```

5.1.2.28 reserve()

```
template<typename T >
void myVector< T >::reserve (
    size_t new_capacity ) [inline]
```

5.1.2.29 resize()

```
template<typename T >
void myVector< T >::resize (
    size_t n ) [inline]
```

5.1.2.30 size()

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

5.1.2.31 swap()

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

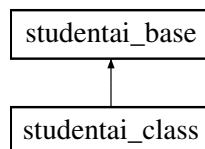
The documentation for this class was generated from the following file:

- [myVector.h](#)

5.2 studentai_base Class Reference

```
#include <funkcijuBazeVektoriai.h>
```

Inheritance diagram for studentai_base:



Public Member Functions

- virtual void [setMasyvas](#) (int a, int b)
- virtual void [testav](#) ()=0
- [studentai_base](#) ()
- virtual [~studentai_base](#) ()

Protected Attributes

- string [vardas_](#)
- string [pavarde_](#)
- [myVector](#)< int > [balai_](#)
- int [egzaminas_](#)
- double [vidurkis_](#)
- double [mediana_](#)
- int * [rodykle_](#)
- int [dydis_](#)

5.2.1 Constructor & Destructor Documentation

5.2.1.1 studentai_base()

```
studentai_base::studentai_base ( ) [inline]
```

5.2.1.2 ~studentai_base()

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

5.2.2 Member Function Documentation

5.2.2.1 setMasyvas()

```
virtual void studentai_base::setMasyvas (
    int a,
    int b ) [inline], [virtual]
```

5.2.2.2 testav()

```
virtual void studentai_base::testav ( ) [pure virtual]
```

Implemented in [studentai_class](#).

5.2.3 Member Data Documentation

5.2.3.1 balai_

```
myVector<int> studentai_base::balai_ [protected]
```

5.2.3.2 dydis_

```
int studentai_base::dydis_ [protected]
```

5.2.3.3 egzaminas_

```
int studentai_base::egzaminas_ [protected]
```

5.2.3.4 mediana_

```
double studentai_base::mediana_ [protected]
```


5.2.3.5 pavarde_

```
string studentai_base::pavarde_ [protected]
```

5.2.3.6 rodykle_

```
int* studentai_base::rodykle_ [protected]
```

5.2.3.7 vardas_

```
string studentai_base::vardas_ [protected]
```

5.2.3.8 vidurkis_

```
double studentai_base::vidurkis_ [protected]
```

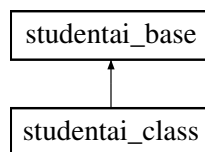
The documentation for this class was generated from the following file:

- [funkcijuBazeVektoriai.h](#)

5.3 studentai_class Class Reference

```
#include <funkcijuBazeVektoriai.h>
```

Inheritance diagram for studentai_class:



Public Member Functions

- [studentai_class](#) ()
- [studentai_class](#) (string vardas, string pavarde, [myVector](#)< int > balai, int egzaminas, double vidurkis, double mediana)
- [~studentai_class](#) ()
- [studentai_class](#) (int dydis)
- void [testav](#) ()
- void [setVardas](#) (string a)
- void [setPavarde](#) (string a)
- void [setBalai](#) ([myVector](#)< int > a)
- void [setEgzaminas](#) (int a)
- void [setVidurkis](#) (double a)
- void [setMediana](#) (double a)
- string [getVardas](#) ()
- string [getPavarde](#) ()
- [myVector](#)< int > [getBalai](#) ()
- int [getEgzaminas](#) ()
- double [getVidurkis](#) ()
- double [getMediana](#) ()
- [studentai_class](#) (const [studentai_class](#) &a)
- [studentai_class](#) operator= (const [studentai_class](#) &a)
- [studentai_class](#) ([studentai_class](#) &&a)
- [studentai_class](#) operator= ([studentai_class](#) &&a)

Public Member Functions inherited from [studentai_base](#)

- virtual void [setMasyvas](#) (int a, int b)
- [studentai_base](#) ()
- virtual [~studentai_base](#) ()

Friends

- ostream & [operator<<](#) (ostream &out, [studentai_class](#) a)
- istream & [operator>>](#) (istream &in, [studentai_class](#) &a)

Additional Inherited Members

Protected Attributes inherited from [studentai_base](#)

- string [vardas_](#)
- string [pavarde_](#)
- [myVector](#)< int > [balai_](#)
- int [egzaminas_](#)
- double [vidurkis_](#)
- double [mediana_](#)
- int * [rodykle_](#)
- int [dydis_](#)

5.3.1 Constructor & Destructor Documentation

5.3.1.1 [studentai_class](#)() [1/5]

```
studentai_class::studentai_class ( )
```

5.3.1.2 [studentai_class](#)() [2/5]

```
studentai_class::studentai_class (
    string vardas,
    string pavarde,
    myVector< int > balai,
    int egzaminas,
    double vidurkis,
    double mediana )
```

5.3.1.3 [~studentai_class](#)()

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

5.3.1.4 [studentai_class](#)() [3/5]

```
studentai_class::studentai_class (
    int dydis )
```

5.3.1.5 studentai_class() [4/5]

```
studentai_class::studentai_class (
    const studentai_class & a )
```

5.3.1.6 studentai_class() [5/5]

```
studentai_class::studentai_class (
    studentai_class && a )
```

5.3.2 Member Function Documentation

5.3.2.1 getBalai()

```
myVector< int > studentai_class::getBalai ( ) [inline]
```

5.3.2.2 getEgzaminas()

```
int studentai_class::getEgzaminas ( ) [inline]
```

5.3.2.3 getMediana()

```
double studentai_class::getMediana ( ) [inline]
```

5.3.2.4 getPavarde()

```
string studentai_class::getPavarde ( ) [inline]
```

5.3.2.5 getVardas()

```
string studentai_class::getVardas ( ) [inline]
```

5.3.2.6 getVidurkis()

```
double studentai_class::getVidurkis ( ) [inline]
```

5.3.2.7 operator=() [1/2]

```
studentai_class studentai_class::operator= (
    const studentai_class & a )
```

5.3.2.8 operator=() [2/2]

```
studentai_class studentai_class::operator= (
    studentai_class && a )
```

5.3.2.9 setBalai()

```
void studentai_class::setBalai (
    myVector< int > a ) [inline]
```

5.3.2.10 setEgzaminas()

```
void studentai_class::setEgzaminas (
    int a ) [inline]
```

5.3.2.11 setMediana()

```
void studentai_class::setMediana (
    double a ) [inline]
```

5.3.2.12 setPavarde()

```
void studentai_class::setPavarde (
    string a ) [inline]
```

5.3.2.13 setVardas()

```
void studentai_class::setVardas (
    string a ) [inline]
```

5.3.2.14 setVidurkis()

```
void studentai_class::setVidurkis (
    double a ) [inline]
```

5.3.2.15 testav()

```
void studentai_class::testav ( ) [virtual]
```

Implements [studentai_base](#).

5.3.3 Friends And Related Symbol Documentation

5.3.3.1 operator<<

```
ostream & operator<< (  
    ostream & out,  
    studentai_class a ) [friend]
```

5.3.3.2 operator>>

```
istream & operator>> (  
    istream & in,  
    studentai_class & a ) [friend]
```

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

- [funkcijuBazeVektoriai.h](#)
- [klasesRealizacija.cpp](#)

Chapter 6

File Documentation

6.1 bendrosFunkcijos.cpp File Reference

```
#include "funkcijuBaze.h"
```

Functions

- string [tarpai](#) (string a, int tarpuDydis)
- string [extraSpace](#) (string a, int b)
- mt19937 [mt](#) (rd())
- uniform_int_distribution< int > [distribution](#) (1, 10)
- void [failoGeneracija](#) ()
- void [laikoSausdinimas](#) (duration< double > readTime, duration< double > sortTime, duration< double > typeTime)

Variables

- random_device [rd](#)

6.1.1 Function Documentation

6.1.1.1 distribution()

```
uniform_int_distribution< int > distribution (  
    1 ,  
    10 )
```

6.1.1.2 extraSpace()

```
string extraSpace (  
    string a,  
    int b )
```

6.1.1.3 failoGeneracija()

```
void failoGeneracija ( )
```

6.1.1.4 laikoSausdinimas()

```
void laikoSausdinimas (
    duration< double > readTime,
    duration< double > sortTime,
    duration< double > typeTime )
```

6.1.1.5 mt()

```
mt19937 mt (
    rd() )
```

6.1.1.6 tarpai()

```
string tarpai (
    string a,
    int tarpuDydis )
```

6.1.2 Variable Documentation

6.1.2.1 rd

```
random_device rd
```

6.2 funkcijosVektoriai.cpp File Reference

```
#include "funkcijuBazeVektoriai.h"
#include "myVector.h"
```


Functions

- void [darbasSuVektoriais](#) ()
- void [vektoriuTestavimas](#) ()
- void [klasiuTestavimas](#) ()
- double [medianosApsk](#) ([myVector](#)< int > a, int egzaminas)
- double [vidurkioApsk](#) ([myVector](#)< int > a, int egzaminas)
- void [pirmasPasirinkimas](#) ()
- void [antrasPasirinkimas](#) ()
- void [treciasPasirinkimas](#) ()
- void [NuskaitymasFailo](#) (string fileName)
- void [spausdinimasFaile](#) ()
- void [spausdinimasTerminale](#) ()
- void [spausdinimasFaileSkirstymas](#) ()
- void [spausdinimasTerminaleSkirstymas](#) ()
- bool [rusiavimasVardas](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- bool [rusiavimasPavarde](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- bool [rusiavimasMediana](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- bool [rusiavimasVidurkis](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- void [rusiavimoMenu](#) ()
- void [rusiavimoMenuSkirstymas](#) ()
- bool [Less](#) ([studentai_class](#) a)
- bool [LessM](#) ([studentai_class](#) a)
- void [skirstymas](#) ()

6.2.1 Function Documentation

6.2.1.1 [antrasPasirinkimas\(\)](#)

```
void antrasPasirinkimas ( )
```

6.2.1.2 [darbasSuVektoriais\(\)](#)

```
void darbasSuVektoriais ( )
```

6.2.1.3 [klasiuTestavimas\(\)](#)

```
void klasiuTestavimas ( )
```

6.2.1.4 [Less\(\)](#)

```
bool Less (
    studentai\_class a )
```

6.2.1.5 [LessM\(\)](#)

```
bool LessM (
    studentai\_class a )
```

6.2.1.6 medianosApsk()

```
double medianosApsk (
    myVector< int > a,
    int egzaminas )
```

6.2.1.7 NuskaitymasFailo()

```
void NuskaitymasFailo (
    string fileName )
```

6.2.1.8 pirmasPasirinkimas()

```
void pirmasPasirinkimas ( )
```

6.2.1.9 rusiavimasMediana()

```
bool rusiavimasMediana (
    studentai_class & a,
    studentai_class & b )
```

6.2.1.10 rusiavimasPavarde()

```
bool rusiavimasPavarde (
    studentai_class & a,
    studentai_class & b )
```

6.2.1.11 rusiavimasVardas()

```
bool rusiavimasVardas (
    studentai_class & a,
    studentai_class & b )
```

6.2.1.12 rusiavimasVidurkis()

```
bool rusiavimasVidurkis (
    studentai_class & a,
    studentai_class & b )
```

6.2.1.13 rusiavimoMenu()

```
void rusiavimoMenu ( )
```

6.2.1.14 rusiavimoMenuSkirstymas()

```
void rusiavimoMenuSkirstymas ( )
```

6.2.1.15 skirstymas()

```
void skirstymas ( )
```

6.2.1.16 spausdinimasFaile()

```
void spausdinimasFaile ( )
```

6.2.1.17 spausdinimasFaileSkirstymas()

```
void spausdinimasFaileSkirstymas ( )
```

6.2.1.18 spausdinimasTerminale()

```
void spausdinimasTerminale ( )
```

6.2.1.19 spausdinimasTerminaleSkirstymas()

```
void spausdinimasTerminaleSkirstymas ( )
```

6.2.1.20 treciasPasirinkimas()

```
void treciasPasirinkimas ( )
```

6.2.1.21 vektoriuTestavimas()

```
void vektoriuTestavimas ( )
```

6.2.1.22 vidurkioApsk()

```
double vidurkioApsk (
    myVector< int > a,
    int egzaminas )
```

6.3 FunkcijuBaze.h File Reference

```
#include "includes.h"
```

Functions

- void [darbasSuVektoriais](#) ()
- void [darbasSuDekais](#) ()
- void [darbasSuListais](#) ()
- string [tarpai](#) (string a, int tarpuDydis)
- string [extraSpace](#) (string a, int b)
- void [failoGeneracija](#) ()

6.3.1 Function Documentation

6.3.1.1 darbasSuDekais()

```
void darbasSuDekais ( )
```

6.3.1.2 darbasSuListais()

```
void darbasSuListais ( )
```

6.3.1.3 darbasSuVektoriais()

```
void darbasSuVektoriais ( )
```

6.3.1.4 extraSpace()

```
string extraSpace (
    string a,
    int b )
```

6.3.1.5 failoGeneracija()

```
void failoGeneracija ( )
```

6.3.1.6 tarpai()

```
string tarpai (
    string a,
    int tarpuDydis )
```

6.4 FunkcijuBaze.h

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

```
00001 #pragma once
00002 #include "includes.h"
00003
00004 //funkcijos main.cpp
00005 void darbasSuVektoriais();
00006 void darbasSuDekais();
00007 void darbasSuListais();
00008
00009 //papildomos funkcijos darbui
00010 string tarpai(string a, int tarpuDydis);
00011 string extraSpace (string a, int b);
00012
00013 void failoGeneracija();
```

6.5 funkcijuBazeVektoriai.h File Reference

```
#include "includes.h"
#include "myVector.h"
```

Classes

- class [studentai_base](#)
- class [studentai_class](#)

Functions

- string [tarpai](#) (string a, int tarpuDydis)
- double [vidurkioApsk](#) ([myVector](#)< int > a, int egzaminas)
- double [medianosApsk](#) ([myVector](#)< int > a, int egzaminas)
- string [extraSpace](#) (string a, int b)
- void [vektoriuTestavimas](#) ()
- void [klasiuTestavimas](#) ()
- void [pirmasPasirinkimas](#) ()
- void [antrasPasirinkimas](#) ()
- void [treciasPasirinkimas](#) ()
- void [NuskaitymasFailo](#) (string fileName)
- void [failoGeneracija](#) ()
- void [skirstymas](#) ()
- void [rusiavimoMenu](#) ()
- void [rusiavimoMenuSkirstymas](#) ()
- bool [rusiavimasVardas](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- bool [rusiavimasPavarde](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- bool [rusiavimasVidurkis](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- bool [rusiavimasMediana](#) ([studentai_class](#) &a, [studentai_class](#) &b)
- void [spausdinimasFaile](#) ()
- void [spausdinimasTerminale](#) ()
- void [spausdinimasFaileSkirstymas](#) ()
- void [spausdinimasTerminaleSkirstymas](#) ()
- void [laikoSpausdinimas](#) (duration< double > readTime, duration< double > sortTime, duration< double > typeTime)

6.5.1 Function Documentation

6.5.1.1 antrasPasirinkimas()

```
void antrasPasirinkimas ( )
```

6.5.1.2 extraSpace()

```
string extraSpace (
    string a,
    int b )
```

6.5.1.3 failoGeneracija()

```
void failoGeneracija ( )
```

6.5.1.4 klasiuTestavimas()

```
void klasiuTestavimas ( )
```

6.5.1.5 laikoSpausdinimas()

```
void laikoSpausdinimas (
    duration< double > readTime,
    duration< double > sortTime,
    duration< double > typeTime )
```

6.5.1.6 medianosApsk()

```
double medianosApsk (
    myVector< int > a,
    int egzaminas )
```

6.5.1.7 NuskaitymasFailo()

```
void NuskaitymasFailo (
    string fileName )
```

6.5.1.8 pirmasPasirinkimas()

```
void pirmasPasirinkimas ( )
```

6.5.1.9 rusiavimasMediana()

```
bool rusiavimasMediana (
    studentai_class & a,
    studentai_class & b )
```

6.5.1.10 rusiavimasPavarde()

```
bool rusiavimasPavarde (
    studentai_class & a,
    studentai_class & b )
```

6.5.1.11 rusiavimasVardas()

```
bool rusiavimasVardas (
    studentai_class & a,
    studentai_class & b )
```

6.5.1.12 rusiavimasVidurkis()

```
bool rusiavimasVidurkis (
    studentai_class & a,
    studentai_class & b )
```

6.5.1.13 rusiavimoMenu()

```
void rusiavimoMenu ( )
```

6.5.1.14 rusiavimoMenuSkirstymas()

```
void rusiavimoMenuSkirstymas ( )
```

6.5.1.15 skirstymas()

```
void skirstymas ( )
```

6.5.1.16 spausdinimasFaile()

```
void spausdinimasFaile ( )
```

6.5.1.17 spausdinimasFaileSkirstymas()

```
void spausdinimasFaileSkirstymas ( )
```

6.5.1.18 spausdinimasTerminale()

```
void spausdinimasTerminale ( )
```

6.5.1.19 spausdinimasTerminaleSkirstymas()

```
void spausdinimasTerminaleSkirstymas ( )
```

6.5.1.20 tarpai()

```
string tarpai (
    string a,
    int tarpuDydis )
```

6.5.1.21 treciasPasirinkimas()

```
void treciasPasirinkimas ( )
```

6.5.1.22 vektoriųTestavimas()

```
void vektoriųTestavimas ( )
```

6.5.1.23 vidurkioApsk()

```
double vidurkioApsk (
    myVector< int > a,
    int egzaminas )
```

6.6 funkcijuBazeVektoriai.h

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

```
00001 #pragma once
00002 #include "includes.h"
00003 #include "myVector.h"
00004 // struct studentai{
00005 //     string vardas;
00006 //     string pavarde;
00007 //     myVector<int> balai;
00008 //     int egzaminas;
00009 //     double vidurkis;
00010 //     double mediana;
00011 // };
00012 string tarpai(string a, int tarpuDydis); // turi buti pirmiau uz klase del perdengto operatoriaus «
00013 double vidurkioApsk(myVector<int> a, int egzaminas);
00014 double medianosApsk(myVector<int> a, int egzaminas);
00015
00016
00017 class studentai_base{
00018     protected:
00019         string vardas_;
00020         string pavarde_;
00021         myVector<int> balai_;
00022         int egzaminas_;
00023         double vidurkis_;
```



```

00024     double  mediana_;
00025     int*    rodykle_;
00026     int     dydis_;
00027     public:
00028         virtual void setMasyvas(int a, int b){rodykle_[a] = b;};
00029         virtual void testav() = 0;
00030         studentai_base(){}; //default konstruktorius
00031         virtual ~studentai_base(){};
00032 };
00033
00034
00035
00036
00037 class studentai_class : public studentai_base{
00038     public:
00039         studentai_class();
00040         studentai_class(string vardas, string pavarde, myVector<int> balai, int egzaminas, double
00041     vidurkis, double mediana);
00042         ~studentai_class(){delete[] rodykle_;};
00043         //testavimui
00044         studentai_class(int dydis);
00045
00046         void testav();
00047         void setVardas(string a){vardas_ = a;};
00048         void setPavarde(string a){pavarde_ = a;};
00049         void setBalai(myVector<int> a){balai_ = a;};
00050         void setEgzaminas(int a){egzaminas_ = a;};
00051         void setVidurkis(double a){vidurkis_ = a;};
00052         void setMediana(double a){mediana_ = a;};
00053
00054         string getVardas(){return vardas_};
00055         string getPavarde(){return pavarde_};
00056         myVector<int> getBalai(){return balai_};
00057         int getEgzaminas(){return egzaminas_};
00058         double getVidurkis(){return vidurkis_};
00059         double getMediana(){return mediana_};
00060         friend ostream& operator<<(ostream& out, studentai_class a); //output overloading
00061         friend istream& operator>>(istream& in, studentai_class &a); //input
00062
00063         studentai_class (const studentai_class& a); // kopiviavimo konstruktorius
00064         studentai_class operator=(const studentai_class& a); // kopiviavimo operatorius / priskirimas
00065         studentai_class (studentai_class&& a); // move konstruktorius
00066         studentai_class operator=(studentai_class&& a); // move operatorius
00067 };
00068 string extraSpace (string a, int b);
00069
00070 void vektoriuTestavimas();
00071 void klasiuTestavimas();
00072 //Vektorius
00073 void pirmasPasirinkimas();
00074 void antrasPasirinkimas();
00075 void treciasPasirinkimas();
00076
00077 void NuskaitymasFailo(string fileName);
00078 void failoGeneracija();
00079
00080 void skirstymas();
00081
00082 void rusiavimoMenu();
00083 void rusiavimoMenuSkirstymas();
00084
00085 bool rusiavimasVardas(studentai_class &a, studentai_class &b);
00086 bool rusiavimasPavarde(studentai_class &a, studentai_class &b);
00087 bool rusiavimasVidurkis(studentai_class &a, studentai_class &b);
00088 bool rusiavimasMediana(studentai_class &a, studentai_class &b);
00089
00090 void spausdinimasFaile();
00091 void spausdinimasTerminale();
00092 void spausdinimasFaileSkirstymas();
00093 void spausdinimasTerminaleSkirstymas();
00094
00095 void laikoSpausdinimas(duration<double> readTime, duration<double> sortTime ,duration<double>
    typeTime);

```

6.7 Includes.h File Reference

```

#include <iostream>
#include <iomanip>
#include <algorithm>

```

```
#include <ctime>
#include <fstream>
#include <chrono>
#include <random>
#include <sstream>
```

Variables

- random_device [rd](#)
- mt19937 [mt](#)
- uniform_int_distribution< int > [distribution](#)

6.7.1 Variable Documentation

6.7.1.1 distribution

```
uniform_int_distribution<int> distribution [extern]
```

6.7.1.2 mt

```
mt19937 mt [extern]
```

6.7.1.3 rd

```
random_device rd [extern]
```

6.8 Includes.h

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

```
00001 #ifndef include
00002 #define include
00003 #include <iostream>
00004 #include <iomanip>
00005 #include <algorithm>
00006 #include <ctime>
00007 #include <fstream>
00008 #include <chrono>
00009 #include <random>
00010 #include <sstream>
00011
00012 using namespace std;
00013 using namespace std::chrono;
00014
00015 extern random_device rd;
00016 extern mt19937 mt;
00017 extern uniform_int_distribution<int> distribution;
00018 #endif
00019 // vidurkio ir medianos funkcijos visos tos pacios, gal reiks pakeisti
```

6.9 klasesRealizacija.cpp File Reference

```
#include "funkcijuBazeVektoriai.h"
#include "myVector.h"
```

Functions

- ostream & [operator<<](#) (ostream &out, [studentai_class](#) a)
- istream & [operator>>](#) (istream &in, [studentai_class](#) &a)

6.9.1 Function Documentation

6.9.1.1 operator<<()

```
ostream & operator<< (  
    ostream & out,  
    studentai\_class a )
```

6.9.1.2 operator>>()

```
istream & operator>> (  
    istream & in,  
    studentai\_class & a )
```

6.10 main.cpp File Reference

```
#include "funkcijuBaze.h"
```

Functions

- int [main](#) ()

6.10.1 Function Documentation

6.10.1.1 main()

```
int main ( )
```

6.11 myVector.h File Reference

```
#include "Includes.h"
```

Classes

- class [myVector< T >](#)

6.12 myVector.h

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

```

00001 #pragma once
00002 #include "Includes.h"
00003 template <typename T> class myVector{
00004     private:
00005         T* arr;
00006         int capacity;
00007         int currentSize;
00008
00009     public:
00010         //creates an empty vector with capacity of 1
00011         myVector(){
00012             arr = new T[1];
00013             capacity = 1;
00014             currentSize = 0;
00015         }
00016
00017         //creates a vector and fills it
00018         myVector(size_t value, T data){
00019             arr = new T[value];
00020             for (int i = 0; i < value; i++){
00021                 arr[i] = data;
00022             }
00023             capacity = value;
00024             currentSize = value;
00025         }
00026         // Initializer list constructor
00027         myVector(std::initializer_list<T> ilist) : arr(new T[1]), capacity(1), currentSize(0) {
00028             reserve(ilist.size());
00029             for (const auto& elem : ilist) {
00030                 push_back(elem);
00031             }
00032         }
00033         myVector(T data){
00034             arr = new T[1];
00035             arr[0] = data;
00036             capacity = 1;
00037             currentSize = 1;
00038         }
00039
00040         // Copy constructor
00041         myVector(const myVector& other) : arr(nullptr), capacity(0), currentSize(0) {
00042             reserve(other.currentSize);
00043             std::copy(other.arr, other.arr + other.currentSize, arr);
00044             currentSize = other.currentSize;
00045         }
00046
00047         // Move constructor
00048         myVector(myVector&& other) noexcept : arr(other.arr), capacity(other.capacity),
00049             currentSize(other.currentSize) {
00050             other.arr = nullptr;
00051             other.capacity = 0;
00052             other.currentSize = 0;
00053         }
00054         // Copy assignment operator
00055         myVector& operator=(const myVector& other) {
00056             if (this == &other) {
00057                 return *this;
00058             }
00059             T* new_arr = new T[other.capacity];
00060             std::copy(other.arr, other.arr + other.currentSize, new_arr);
00061             delete[] arr;
00062             arr = new_arr;
00063             capacity = other.capacity;
00064             currentSize = other.currentSize;
00065             return *this;
00066         }
00067
00068         // Move assignment operator
00069         myVector& operator=(myVector&& other) noexcept {
00070             if (this == &other) {
00071                 return *this;
00072             }
00073             delete[] arr;
00074             arr = other.arr;
00075             capacity = other.capacity;
00076             currentSize = other.currentSize;
00077             other.arr = nullptr;
00078             other.capacity = 0;
00079             other.currentSize = 0;
00080             return *this;
00081         }

```

```

00082
00083 ~myVector(){
00084     delete [] arr;
00085 }
00086 void reserve(size_t new_capacity) {
00087     if (new_capacity > capacity) {
00088         T* new_arr = new T[new_capacity];
00089         for (size_t i = 0; i < currentSize; ++i) {
00090             new_arr[i] = arr[i];
00091         }
00092         delete[] arr;
00093         arr = new_arr;
00094         capacity = new_capacity;
00095     }
00096 }
00097 void push_back(T duomenys){
00098
00099     // if capacity reached, double size
00100     if (currentSize == capacity) {
00101         T* temp = new T[2 * capacity];
00102
00103         // copy elements
00104         for (int i = 0; i < capacity; i++) {
00105             temp[i] = arr[i];
00106         }
00107
00108         // deleting previous array
00109         delete[] arr;
00110         capacity *= 2;
00111         arr = temp;
00112     }
00113
00114     // Adding data
00115     arr[currentSize] = duomenys;
00116     currentSize++;
00117 }
00118
00119 // function to delete last element
00120 void pop_back(){
00121     currentSize--;
00122 }
00123
00124
00125 T at(size_t index){
00126     if (index < currentSize){
00127         return arr[index];
00128     }
00129     else{
00130         throw std::out_of_range("Vector out of range");
00131     }
00132 }
00133
00134
00135 // function to get size of the vector
00136 int size() const{
00137     return currentSize;
00138 }
00139 // function to get capacity of the vector
00140 int getCapacity(){
00141     return capacity;
00142 }
00143
00144
00145 // function to print out all array elements
00146 void print()
00147 {
00148     for (int i = 0; i < currentSize; i++) {
00149         cout << arr[i] << " ";
00150     }
00151     cout << endl;
00152 }
00153 // check if empty
00154 bool empty(){
00155     if (currentSize == 0){
00156         return true;
00157     }else{
00158         return false;
00159     }
00160 }
00161
00162 // clears vector
00163 void clear(){
00164     currentSize = 0;
00165 }
00166
00167
00168 // Access the first element

```

```

00169     T& first() {
00170         if (currentSize == 0) {
00171             throw std::out_of_range("Vector is empty");
00172         }
00173         return arr[0];
00174     }
00175
00176     // Access the last element
00177     T& last() {
00178         if (currentSize == 0) {
00179             throw std::out_of_range("Vector is empty");
00180         }
00181         return arr[currentSize - 1];
00182     }
00183
00184     // Iterator to the beginning
00185     T* begin() {
00186         return arr;
00187     }
00188
00189     // Iterator to the end
00190     T* end() {
00191         return arr + currentSize;
00192     }
00193
00194
00195
00196     T& operator[] (size_t index) const{
00197         if (index >= capacity) {
00198             throw std::out_of_range("Index out of range");
00199         }
00200         return arr[index];
00201     }
00202
00203     // Assign function to assign multiple copies of a value
00204     void assign( size_t count, const T& value ){
00205         clear();
00206         for (int i = 0; i < count; i++){
00207             arr[i] = value;
00208         }
00209
00210     // Assign function to assign a range of elements
00211     template <typename InputIterator>
00212     void assign(InputIterator first, InputIterator last) {
00213         size_t new_size = std::distance(first, last);
00214         if (new_size > capacity) {
00215             reserve(new_size);
00216         }
00217         currentSize = new_size;
00218         std::copy(first, last, arr);
00219     }
00220
00221     // Erase a single element at position
00222     void erase(T* position) {
00223         if (position < arr || position >= arr + currentSize) {
00224             throw std::out_of_range("Position out of range");
00225         }
00226         std::copy(position + 1, arr + currentSize, position);
00227         --currentSize;
00228     }
00229
00230     // Erase elements in range
00231     void erase(T* first, T* last) {
00232         if (first < arr || last > arr + currentSize || first > last) {
00233             throw std::out_of_range("Range out of range");
00234         }
00235         std::copy(last, arr + currentSize, first);
00236         currentSize -= (last - first);
00237     }
00238
00239
00240     // Swap the contents with another vector
00241     void swap(myVector& other) {
00242         std::swap(arr, other.arr);
00243         std::swap(capacity, other.capacity);
00244         std::swap(currentSize, other.currentSize);
00245     }
00246
00247     // Resize the vector to contain n elements
00248     void resize(size_t n) {
00249         if (n > capacity) {
00250             reserve(n);
00251         }
00252         if (n > currentSize) {
00253             std::fill(arr + currentSize, arr + n, T());
00254         }
00255         currentSize = n;

```

```

00256     }
00257     // Insert a single element at position
00258     T* insert(T* position, const T& val) {
00259         size_t index = position - arr;
00260         if (currentSize == capacity) {
00261             reserve(capacity == 0 ? 1 : 2 * capacity);
00262         }
00263         position = arr + index;
00264         std::copy_backward(position, arr + currentSize, arr + currentSize + 1);
00265         *position = val;
00266         ++currentSize;
00267         return position;
00268     }
00269
00270     // Insert multiple elements at position
00271     void insert(T* position, size_t n, const T& val) {
00272         size_t index = position - arr;
00273         if (currentSize + n > capacity) {
00274             reserve(currentSize + n);
00275         }
00276         position = arr + index;
00277         std::copy_backward(position, arr + currentSize, arr + currentSize + n);
00278         std::fill(position, position + n, val);
00279         currentSize += n;
00280     }
00281
00282     // Insert elements from range [first, last) at position
00283     template <typename InputIterator>
00284     void insert(T* position, InputIterator first, InputIterator last) {
00285         size_t index = position - arr;
00286         size_t n = std::distance(first, last);
00287         if (currentSize + n > capacity) {
00288             reserve(currentSize + n);
00289         }
00290         position = arr + index;
00291         std::copy_backward(position, arr + currentSize, arr + currentSize + n);
00292         std::copy(first, last, position);
00293         currentSize += n;
00294     }
00295
00296     // Comparison operators
00297     bool operator==(const myVector& other) const {
00298         if (currentSize != other.currentSize) return false;
00299         for (size_t i = 0; i < currentSize; ++i) {
00300             if (arr[i] != other.arr[i]) return false;
00301         }
00302         return true;
00303     }
00304
00305     bool operator!=(const myVector& other) const {
00306         return !(*this == other);
00307     }
00308
00309     bool operator<(const myVector& other) const {
00310         return std::lexicographical_compare(arr, arr + currentSize, other.arr, other.arr +
00311 other.currentSize);
00312     }
00313
00314     bool operator<=(const myVector& other) const {
00315         return !(other < *this);
00316     }
00317
00318     bool operator>(const myVector& other) const {
00319         return other < *this;
00320     }
00321
00322     bool operator>=(const myVector& other) const {
00323         return !(*this < other);
00324     }
00325 };

```

6.13 readMe.md File Reference

Index

- ~myVector
 - myVector< T >, [10](#)
- ~studentai_base
 - studentai_base, [16](#)
- ~studentai_class
 - studentai_class, [18](#)
- antrasPasirinkimas
 - funkcijosVektoriai.cpp, [25](#)
 - funkcijuBazeVektoriai.h, [30](#)
- assign
 - myVector< T >, [11](#)
- at
 - myVector< T >, [11](#)
- balai_
 - studentai_base, [16](#)
- begin
 - myVector< T >, [11](#)
- bendrosFunkcijos.cpp, [23](#)
 - distribution, [23](#)
 - extraSpace, [23](#)
 - failoGeneracija, [23](#)
 - laikoSpausdinimas, [24](#)
 - mt, [24](#)
 - rd, [24](#)
 - tarpai, [24](#)
- clear
 - myVector< T >, [11](#)
- darbasSuDekais
 - FunkcijuBaze.h, [28](#)
- darbasSuListais
 - FunkcijuBaze.h, [28](#)
- darbasSuVektoriais
 - funkcijosVektoriai.cpp, [25](#)
 - FunkcijuBaze.h, [28](#)
- distribution
 - bendrosFunkcijos.cpp, [23](#)
 - Includes.h, [34](#)
- dydis_
 - studentai_base, [16](#)
- egzaminas_
 - studentai_base, [16](#)
- empty
 - myVector< T >, [11](#)
- end
 - myVector< T >, [11](#)
- erase
 - myVector< T >, [12](#)
- extraSpace
 - bendrosFunkcijos.cpp, [23](#)
 - FunkcijuBaze.h, [28](#)
 - funkcijuBazeVektoriai.h, [30](#)
- failoGeneracija
 - bendrosFunkcijos.cpp, [23](#)
 - FunkcijuBaze.h, [28](#)
 - funkcijuBazeVektoriai.h, [30](#)
- first
 - myVector< T >, [12](#)
- funkcijosVektoriai.cpp, [24](#)
 - antrasPasirinkimas, [25](#)
 - darbasSuVektoriais, [25](#)
 - klasiuTestavimas, [25](#)
 - Less, [25](#)
 - LessM, [25](#)
 - medianosApsk, [25](#)
 - NuskaitymasFailo, [26](#)
 - pirmasPasirinkimas, [26](#)
 - rusiavimasMediana, [26](#)
 - rusiavimasPavarde, [26](#)
 - rusiavimasVardas, [26](#)
 - rusiavimasVidurkis, [26](#)
 - rusiavimoMenu, [26](#)
 - rusiavimoMenuSkirstymas, [26](#)
 - skirstymas, [27](#)
 - spausdinimasFaile, [27](#)
 - spausdinimasFaileSkirstymas, [27](#)
 - spausdinimasTerminale, [27](#)
 - spausdinimasTerminaleSkirstymas, [27](#)
 - treciasPasirinkimas, [27](#)
 - vektoriuTestavimas, [27](#)
 - vidurkioApsk, [27](#)
- FunkcijuBaze.h, [27](#)
 - darbasSuDekais, [28](#)
 - darbasSuListais, [28](#)
 - darbasSuVektoriais, [28](#)
 - extraSpace, [28](#)
 - failoGeneracija, [28](#)
 - tarpai, [28](#)
- funkcijuBazeVektoriai.h, [29](#)
 - antrasPasirinkimas, [30](#)
 - extraSpace, [30](#)
 - failoGeneracija, [30](#)
 - klasiuTestavimas, [30](#)
 - laikoSpausdinimas, [30](#)
 - medianosApsk, [30](#)
 - NuskaitymasFailo, [30](#)

- pirmasPasirinkimas, 30
 - rusiavimasMediana, 30
 - rusiavimasPavarde, 31
 - rusiavimasVardas, 31
 - rusiavimasVidurkis, 31
 - rusiavimoMenu, 31
 - rusiavimoMenuSkirstymas, 31
 - skirstymas, 31
 - spausdinimasFaile, 31
 - spausdinimasFaileSkirstymas, 31
 - spausdinimasTerminale, 31
 - spausdinimasTerminaleSkirstymas, 32
 - tarpai, 32
 - treciasPasirinkimas, 32
 - vektoriuTestavimas, 32
 - vidurkioApsk, 32
- getBalai
 - studentai_class, 19
- getCapacity
 - myVector< T >, 12
- getEgzaminas
 - studentai_class, 19
- getMediana
 - studentai_class, 19
- getPavarde
 - studentai_class, 19
- getVardas
 - studentai_class, 19
- getVidurkis
 - studentai_class, 19
- Includes.h, 33
 - distribution, 34
 - mt, 34
 - rd, 34
- insert
 - myVector< T >, 12
- klasesRealizacija.cpp, 34
 - operator<<, 35
 - operator>>, 35
- klasiuTestavimas
 - funkcijosVektoriai.cpp, 25
 - funkcijuBazeVektoriai.h, 30
- laikoSausdinimas
 - bendrosFunkcijos.cpp, 24
 - funkcijuBazeVektoriai.h, 30
- last
 - myVector< T >, 13
- Less
 - funkcijosVektoriai.cpp, 25
- LessM
 - funkcijosVektoriai.cpp, 25
- main
 - main.cpp, 35
- main.cpp, 35
 - main, 35
- mediana_
 - studentai_base, 16
- medianosApsk
 - funkcijosVektoriai.cpp, 25
 - funkcijuBazeVektoriai.h, 30
- mt
 - bendrosFunkcijos.cpp, 24
 - Includes.h, 34
- myVector
 - myVector< T >, 10
- myVector< T >, 9
 - ~myVector, 10
 - assign, 11
 - at, 11
 - begin, 11
 - clear, 11
 - empty, 11
 - end, 11
 - erase, 12
 - first, 12
 - getCapacity, 12
 - insert, 12
 - last, 13
 - myVector, 10
 - operator!=, 13
 - operator<, 13
 - operator<=, 13
 - operator>, 14
 - operator>=, 14
 - operator=, 13
 - operator==, 13
 - operator[], 14
 - pop_back, 14
 - print, 14
 - push_back, 14
 - reserve, 14
 - resize, 14
 - size, 15
 - swap, 15
- myVector.h, 35
- NuskaitymasFailo
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 30
- operator!=
 - myVector< T >, 13
- operator<
 - myVector< T >, 13
- operator<<
 - klasesRealizacija.cpp, 35
 - studentai_class, 21
- operator<=
 - myVector< T >, 13
- operator>
 - myVector< T >, 14
- operator>>
 - klasesRealizacija.cpp, 35

- studentai_class, 21
- operator>=
 - myVector< T >, 14
- operator=
 - myVector< T >, 13
 - studentai_class, 19
- operator==
 - myVector< T >, 13
- operator[]
 - myVector< T >, 14
- pavarde_
 - studentai_base, 16
- pirmasPasirinkimas
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 30
- pop_back
 - myVector< T >, 14
- print
 - myVector< T >, 14
- push_back
 - myVector< T >, 14
- rd
 - bendrosFunkcijos.cpp, 24
 - Includes.h, 34
- readMe.md, 39
- reserve
 - myVector< T >, 14
- resize
 - myVector< T >, 14
- rodykle_
 - studentai_base, 17
- rusiavimasMediana
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 31
- rusiavimasPavarde
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 31
- rusiavimasVardas
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 31
- rusiavimasVidurkis
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 31
- rusiavimoMenu
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 31
- rusiavimoMenuSkirstymas
 - funkcijosVektoriai.cpp, 26
 - funkcijuBazeVektoriai.h, 31
- setBalai
 - studentai_class, 20
- setEgzaminas
 - studentai_class, 20
- setMasyvas
 - studentai_base, 16
- setMediana
 - studentai_class, 20
- setPavarde
 - studentai_class, 20
- setVardas
 - studentai_class, 20
- setVidurkis
 - studentai_class, 20
- size
 - myVector< T >, 15
- skirstymas
 - funkcijosVektoriai.cpp, 27
 - funkcijuBazeVektoriai.h, 31
- spausdinimasFaile
 - funkcijosVektoriai.cpp, 27
 - funkcijuBazeVektoriai.h, 31
- spausdinimasFaileSkirstymas
 - funkcijosVektoriai.cpp, 27
 - funkcijuBazeVektoriai.h, 31
- spausdinimasTerminale
 - funkcijosVektoriai.cpp, 27
 - funkcijuBazeVektoriai.h, 31
- spausdinimasTerminaleSkirstymas
 - funkcijosVektoriai.cpp, 27
 - funkcijuBazeVektoriai.h, 32
- studentai_base, 15
 - ~studentai_base, 16
 - balai_, 16
 - dydis_, 16
 - egzaminas_, 16
 - mediana_, 16
 - pavarde_, 16
 - rodykle_, 17
 - setMasyvas, 16
 - studentai_base, 16
 - testav, 16
 - vardas_, 17
 - vidurkis_, 17
- studentai_class, 17
 - ~studentai_class, 18
 - getBalai, 19
 - getEgzaminas, 19
 - getMediana, 19
 - getPavarde, 19
 - getVardas, 19
 - getVidurkis, 19
 - operator<<, 21
 - operator>>, 21
 - operator=, 19
 - setBalai, 20
 - setEgzaminas, 20
 - setMediana, 20
 - setPavarde, 20
 - setVardas, 20
 - setVidurkis, 20
 - studentai_class, 18, 19
 - testav, 20
- swap
 - myVector< T >, 15

tarpai
 bendrosFunkcijos.cpp, [24](#)
 FunkcijuBaze.h, [28](#)
 funkcijuBazeVektoriai.h, [32](#)
testav
 studentai_base, [16](#)
 studentai_class, [20](#)
treciasPasirinkimas
 funkcijosVektoriai.cpp, [27](#)
 funkcijuBazeVektoriai.h, [32](#)

V3 Versija, [1](#)
vardas_
 studentai_base, [17](#)
vektoriuTestavimas
 funkcijosVektoriai.cpp, [27](#)
 funkcijuBazeVektoriai.h, [32](#)
vidurkioApsk
 funkcijosVektoriai.cpp, [27](#)
 funkcijuBazeVektoriai.h, [32](#)
vidurkis_
 studentai_base, [17](#)