My Project

Generated by Doxygen 1.10.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

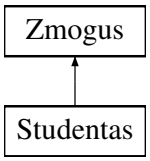
Chapter 4

Class Documentation

4.1 Studentas Class Reference

`#include <studentas.h>`

Inheritance diagram for Studentas:



Public Member Functions

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

Public Member Functions inherited from Zmogus

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

Private Attributes

- Vector< int > pazymiai_
- int egzamino_rezultatas_
- double galutinis_vid_
- double mediana_

Friends

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

Additional Inherited Members

Protected Attributes inherited from Zmogus

- string vardas_
- string pavarde_

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Studentas() [1/4]

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

4.1.1.2 Studentas() [2/4]

```
Studentas::Studentas (
            const Vector< int > & pazymiai,
            const string & vardas,
            const string & pavarde,
            int egzamino_rezultatas,
            double galutinis_vid,
            double mediana )  [inline]
```

4.1.1.3 Studentas() [3/4]

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

4.1.1.4 Studentas() [4/4]

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

4.1.1.5 ~Studentas()

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

4.1.2 Member Function Documentation

4.1.2.1 addPazymys()

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

4.1.2.2 getEgzaminoRezultatas()

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

4.1.2.3 getGalutinisVid()

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

4.1.2.4 getMediana()

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

4.1.2.5 getPavarde()

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

Implements Zmogus.

4.1.2.6 getPazymiai()

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

4.1.2.7 getVardas()

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

Implements Zmogus.

4.1.2.8 operator=() [1/2]

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

4.1.2.9 operator=() [2/2]

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

4.1.2.10 setEgzaminoRezultatas()

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

4.1.2.11 setGalutinisVid()

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

4.1.2.12 setMediana()

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

4.1.2.13 setPavarde()

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

Reimplemented from Zmogus.

4.1.2.14 setPazymiai()

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

4.1.2.15 setVardas()

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

Reimplemented from Zmogus.

4.1.3 Friends And Related Symbol Documentation

4.1.3.1 operator<<

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

4.1.3.2 operator>>

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

4.1.4 Member Data Documentation

4.1.4.1 egzamino_rezultatas_

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

4.1.4.2 galutinis_vid_

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

4.1.4.3 mediana_

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

4.1.4.4 pazymiai_

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

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

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

4.2 Vector< T > Class Template Reference

Klasė, kuri reprezentuoja dinaminį masyvą.

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

Public Types

- typedef size_t size_type
- typedef T value_type
- typedef T & reference
- typedef const T & const_reference
- typedef T * iterator
- typedef const T * const_iterator

Public Member Functions

- Vector ()

  Konstruktoriai /default.
- Vector (size_type n, const T &t=T{})
- Vector (const Vector &v)
- template<class InputIterator >
  Vector (InputIterator first, InputIterator last)
- Vector (Vector &&v)
- Vector (const std::initializer_list< T > il)
- ~Vector ()

  Destruktorius.
- Vector & operator= (const Vector &other)

  Operator = /copy assignment.
- Vector & operator= (Vector &&other)
- template<class InputIterator >
  void assign (InputIterator first, InputIterator last)

  Assign.
- void assign (size_type n, const value_type &val)
- void assign (std::initializer_list< value_type > il)
- const_reference at (size_type n) const
- T & operator[ ] (size_type n)
- const T & operator[ ] (size_type n) const
- reference at (size_type n)
- reference front ()
- const_reference front () const
- reference back ()
- const_reference back () const
- value_type * data () noexcept
- const value_type * data () const noexcept
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- size_type size () const
- size_type max_size () const
- void resize (size_type sz)
- void resize (size_type sz, const value_type &value)
- size_type capacity () const
- bool empty () const noexcept
- void reserve (size_type n)
- void shrink_to_fit ()
- void clear () noexcept
- iterator insert (const_iterator position, const value_type &val)

- iterator insert (iterator position, size_type n, const value_type &val)
- iterator erase (iterator position)
- iterator erase (iterator first, iterator last)
- void push_back (const value_type &t)
- void push_back (value_type &&val)
- void pop_back ()
- void swap (Vector &x)
- bool operator== (const Vector< T > &other) const
- bool operator!= (const Vector< T > &other) const
- bool operator< (const Vector< T > &other) const
- bool operator<= (const Vector< T > &other) const
- bool operator> (const Vector< T > &other) const
- bool operator>= (const Vector< T > &other) const
- void swap (Vector< T > &x, Vector< T > &y)

Private Member Functions

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

Private Attributes

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

4.2.1 Detailed Description

template<typename T>
class Vector< T >

Klasė, kuri reprezentuoja dinaminį masyvą.

Template Parameters

T	Tipo elementai, saugomi masyve.

4.2.2 Member Typedef Documentation

4.2.2.1 const_iterator

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

Konstantos iteratoriaus tipo narys.

4.2.2.2 const_reference

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

Konstantos nuorodos tipo narys.

4.2.2.3 iterator

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

Iteratoriaus tipo narys.

4.2.2.4 reference

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

Nuorodos tipo narys.

4.2.2.5 size_type

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

Dydžio tipo narys.

4.2.2.6 value_type

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

Reikšmės tipo narys.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 Vector() [1/6]

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

Konstruktoriai /default.

4.2.3.2 Vector() [2/6]

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

4.2.3.3 Vector() [3/6]

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

4.2.3.4 Vector() [4/6]

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

4.2.3.5 Vector() [5/6]

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

4.2.3.6 Vector() [6/6]

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

4.2.3.7 ~Vector()

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

Destruktorius.

4.2.4 Member Function Documentation

4.2.4.1 assign() [1/3]

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

Assign.

4.2.4.2 assign() [2/3]

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

4.2.4.3 assign() [3/3]

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

4.2.4.4 at() [1/2]

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

4.2.4.5 at() [2/2]

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

4.2.4.6 back() [1/2]

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

4.2.4.7 back() [2/2]

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

4.2.4.8 begin() [1/2]

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

4.2.4.9 begin() [2/2]

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

4.2.4.10 capacity()

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

4.2.4.11 clear()

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

4.2.4.12 create() [1/3]

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

4.2.4.13 create() [2/3]

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

4.2.4.14 create() [3/3]

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

4.2.4.15 data() [1/2]

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

4.2.4.16 data() [2/2]

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

4.2.4.17 empty()

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

4.2.4.18 end() [1/2]

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

4.2.4.19 end() [2/2]

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

4.2.4.20 erase() [1/2]

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

4.2.4.21 erase() [2/2]

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

4.2.4.22 front() [1/2]

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

4.2.4.23 front() [2/2]

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

4.2.4.24 grow()

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

4.2.4.25 insert() [1/2]

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

4.2.4.26 insert() [2/2]

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

4.2.4.27 max_size()

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

4.2.4.28 operator"!=()

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

4.2.4.29 operator<()

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

4.2.4.30 operator<=()

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

4.2.4.31 operator=() [1/2]

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

Operator = /copy assignment.

4.2.4.32 operator=() [2/2]

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

4.2.4.33 operator==()

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

4.2.4.34 operator>()

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

4.2.4.35 operator>=()

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

4.2.4.36 operator[]() [1/2]

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

4.2.4.37 operator[]() [2/2]

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

4.2.4.38 pop_back()

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

4.2.4.39 push_back() [1/2]

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

4.2.4.40 push_back() [2/2]

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

4.2.4.41 reserve()

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

4.2.4.42 resize() [1/2]

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

4.2.4.43 resize() [2/2]

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

4.2.4.44 shrink_to_fit()

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

4.2.4.45 size()

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

4.2.4.46 swap() [1/2]

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

4.2.4.47 swap() [2/2]

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

4.2.4.48 unchecked_append()

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

4.2.4.49 uncreate()

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

4.2.5 Member Data Documentation

4.2.5.1 alloc

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

4.2.5.2 avail

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

4.2.5.3 dat

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

4.2.5.4 limit

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

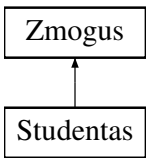
The documentation for this class was generated from the following file:

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

4.3 Zmogus Class Reference

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

Inheritance diagram for Zmogus:



Public Member Functions

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

Protected Attributes

- string vardas_
- string pavarde_

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Zmogus() [1/2]

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

4.3.1.2 Zmogus() [2/2]

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

4.3.1.3 ~Zmogus()

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

4.3.2 Member Function Documentation

4.3.2.1 getPavarde()

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

Implemented in Studentas.

4.3.2.2 getVardas()

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

Implemented in Studentas.

4.3.2.3 setPavarde()

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

Reimplemented in Studentas.

4.3.2.4 setVardas()

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

Reimplemented in Studentas.

4.3.3 Member Data Documentation

4.3.3.1 pavarde_

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

4.3.3.2 vardas_

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

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

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

Chapter 5

File Documentation

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

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

Functions

- double mediana (Vector< int > pazymiai, int egzamino_rezultatas)
- double vidurkis_galutinis (double suma, int n, int egzamino_rezultatas)
- bool palyginti_pagal_varda (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_pavarde (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_mediana (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_galutini_vidurki (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_galutini_vidurki_didejimo_tvarka (const Studentas &a, const Studentas &b)
- bool ar_vargsiukas (const Studentas &student)
- void RezultatuVaizdavimas (const Vector< Studentas > &studentai, int pasirinkimas1)
- void generuotiFaila (string failoPavadinimas, int ndSkaicius, int studentuSkaicius)
- void failoIsvedimasVector1 (const Vector< Studentas > &studentai, string failoPavadinimas)
- void nuskaitytiFailaVector1 (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailo↩ Pavadinimas)
- void failoIsvedimasList1 (const list< Studentas > &studentai, string failoPavadinimas)
- void nuskaitytiFailaList1 (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailo↩ Pavadinimas)
- void failoIsvedimasDeque1 (const deque< Studentas > &studentai, string failoPavadinimas)
- void nuskaitytiFailaDeque1 (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailo↩ Pavadinimas)

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

5.1.1 Function Documentation

5.1.1.1 ar_vargsiukas()

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

5.1.1.2 failoIsvedimasDeque1()

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

5.1.1.3 failoIsvedimasList1()

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

5.1.1.4 failoIsvedimasVector1()

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

5.1.1.5 generuotiFaila()

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

5.1.1.6 mediana()

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

5.1.1.7 nuskaitytiFailaDeque1()

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

5.1.1.8 nuskaitytiFailaDeque2()

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

5.1.1.9 nuskaitytiFailaDeque3()

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

5.1.1.10 nuskaitytiFailaList1()

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

5.1.1.11 nuskaitytiFailaList2()

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

5.1.1.12 nuskaitytiFailaList3()

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

5.1.1.13 nuskaitytiFailaVector1()

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

5.1.1.14 nuskaitytiFailaVector2()

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

5.1.1.15 nuskaitytiFailaVector3()

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

5.1.1.16 palyginti_pagal_galutini_vidurki()

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

5.1.1.17 palyginti_pagal_galutini_vidurki_didejimo_tvarka()

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

5.1.1.18 palyginti_pagal_mediana()

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

5.1.1.19 palyginti_pagal_pavarde()

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

5.1.1.20 palyginti_pagal_varda()

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

5.1.1.21 RezultatuVaizdavimas()

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

5.1.1.22 testai()

```
void testai ( )
```

5.1.1.23 vidurkis_galutinis()

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

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

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

Functions

- double mediana (Vector< int > pazymiai, int egzamino_rezultatas)
- double vidurkis_galutinis (double suma, int n, int egzamino_rezultatas)
- bool palyginti_pagal_varda (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_pavarde (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_mediana (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_galutini_vidurki (const Studentas &a, const Studentas &b)
- bool palyginti_pagal_galutini_vidurki_didejimo_tvarka (const Studentas &a, const Studentas &b)
- bool ar_vargsiukas (const Studentas &student)
- void RezultatuVaizdavimas (const Vector< Studentas > &studentai, int pasirinkimas1)
- void generuotiFaila (string failoPavadinimas, int ndSkaicius, int studentuSkaicius)
- void failoIsvedimasVector1 (const Vector< Studentas > &studentai, string failoPavadinimas)
- void nuskaitytiFailaVector1 (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailo↩ Pavadinimas)
- void failoIsvedimasList1 (const list< Studentas > &studentai, string failoPavadinimas)
- void nuskaitytiFailaList1 (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailo↩ Pavadinimas)
- void failoIsvedimasDeque1 (const deque< Studentas > &studentai, string failoPavadinimas)
- void nuskaitytiFailaDeque1 (string failoPavadinimas, string vargsiukuFailoPavadinimas, string kietakiuFailo↩ Pavadinimas)
- void nuskaitytiFailaVector2 (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void nuskaitytiFailaList2 (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void nuskaitytiFailaDeque2 (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void nuskaitytiFailaVector3 (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void nuskaitytiFailaList3 (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void nuskaitytiFailaDeque3 (string failoPavadinimas, string vargsiukuFailoPavadinimas)
- void testai ()

5.2.1 Function Documentation

5.2.1.1 ar_vargsiukas()

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

5.2.1.2 failoIsvedimasDeque1()

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

5.2.1.3 failoIsvedimasList1()

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

5.2.1.4 failoIsvedimasVector1()

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

5.2.1.5 generuotiFaila()

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

5.2.1.6 mediana()

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

5.2.1.7 nuskaitytiFailaDeque1()

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

5.2.1.8 nuskaitytiFailaDeque2()

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

5.2.1.9 nuskaitytiFailaDeque3()

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

5.2.1.10 nuskaitytiFailaList1()

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

5.2.1.11 nuskaitytiFailaList2()

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

5.2.1.12 nuskaitytiFailaList3()

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

5.2.1.13 nuskaitytiFailaVector1()

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

5.2.1.14 nuskaitytiFailaVector2()

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

5.2.1.15 nuskaitytiFailaVector3()

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

5.2.1.16 palyginti_pagal_galutini_vidurki()

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

5.2.1.17 palyginti_pagal_galutini_vidurki_didejimo_tvarka()

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

5.2.1.18 palyginti_pagal_mediana()

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

5.2.1.19 palyginti_pagal_pavarde()

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

5.2.1.20 palyginti_pagal_varda()

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

5.2.1.21 RezultatuVaizdavimas()

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

5.2.1.22 testai()

```
void testai ( )
```

5.2.1.23 vidurkis_galutinis()

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

5.3 funkcijos.h

Go to the documentation of this file.

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

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

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

Classes

- class Zmogus
- class Studentas

5.5 studentas.h

Go to the documentation of this file.

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

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

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

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

Classes

- class Vector< T >

  Klasė, kuri reprezentuoja dinaminį masyvą.

5.7 vector.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <memory>
00005 #include <algorithm>
00006 #include <limits>
00007
00014 template <typename T>
00015 class Vector{
00016     public:
00017
00018     // MEMBER TYPE
00019     typedef size_t size_type;
00020     typedef T value_type;
00021     typedef T& reference;
00022     typedef const T& const_reference;
00023     typedef T* iterator;
00024     typedef const T* const_iterator;
00026     // MEMBER FUNCTIONS
00027
00030         Vector() {create();}
00032         explicit Vector(size_type n, const T& t = T{}) { create (n,t); }
00034         Vector(const Vector& v) { create(v.begin(), v.end()); }
00036         template <class InputIterator>
00037         Vector (InputIterator first, InputIterator last) { create(first,last); }
00039         Vector (Vector&& v) {
00040             create();
00041             swap(v);
00042             v.uncreate();
00043         }
00045         Vector(const std::initializer_list<T> il) { create(il.begin(), il.end()); }
00046
00048         ~Vector() {uncreate();}
00049
00052         Vector& operator = (const Vector& other) {
00053             if (this != &other) {
00054                 uncreate();
00055                 create(other.begin(), other.end());
00056             }
00057             return *this;
00058         };
00059
00061         Vector& operator = (Vector&& other) {
00062             if (this != &other) {
00063                 create(other.begin(), other.end());
00064                 uncreate();
00065             }
00066             return *this;
00067         }
00068
00070         template <class InputIterator>
00071         void assign (InputIterator first, InputIterator last) {
00072             uncreate();
00073             create(first, last);
00074         }
00075         void assign (size_type n, const value_type& val) {
00076             uncreate();
00077             create(n, val);
00078         }
00079         void assign (std::initializer_list<value_type> il) {
00080             uncreate();
00081             create(il);
00082         }
00083
00084         // Element access
00085         const_reference at (size_type n) const {
00086             if (n >= size() || n < 0)
00087                 throw std::out_of_range("Index out of range");
00088             return dat[n];
00089         }
00090
00091         T& operator[] (size_type n) {return dat[n];}
00092         const T& operator[] (size_type n) const {return dat[n];}
00093         reference at (size_type n) {
00094             if (n >= size() || n < 0)
00095                 throw std::out_of_range("Index out of range");
00096             return dat[n];
00097         }
00098
00099         reference front() {
00100             return dat[0];
00101         };
00102
00103         const_reference front() const {
00104             return dat[0];
00105         }
00106
```

```
00107         reference back() {
00108             return dat[size() - 1];
00109         }
00110 
00111         const_reference back() const {
00112             return dat[size() - 1];
00113         }
00114 
00115         value_type* data() noexcept {
00116             return dat;
00117         }
00118 
00119         const value_type* data() const noexcept {
00120             return dat;
00121         }
00122 
00123     // Iterators
00124         iterator begin() {return dat;}
00125         const_iterator begin() const {return dat;}
00126         iterator end() {return avail;}
00127         const_iterator end() const {return avail;}
00128 
00129     // Capacity
00130         size_type size() const {return avail-dat;}
00131         size_type max_size() const {return std::numeric_limits<size_type>::max();}
00132         void resize(size_type sz) {
00133             if (sz < size()) {
00134                 iterator it = dat + sz;
00135                 while (it != avail) {
00136                     alloc.destroy(it++);
00137                 }
00138                 avail = dat + sz;
00139             }
00140             else if (sz > capacity()) {
00141                 grow(sz);
00142                 std::uninitialized_fill(avail, dat + sz, value_type());
00143                 avail = dat + sz;
00144             }
00145             else if (sz > size()) {
00146                 std::uninitialized_fill(avail, dat + sz, value_type());
00147                 avail = dat + sz;
00148             }
00149         }
00150 
00151         void resize(size_type sz, const value_type& value) {
00152             if (sz > capacity()) {
00153                 grow(sz);
00154             }
00155 
00156             if (sz > size()) {
00157                 insert(end(), sz - size(), value);
00158             } else if (sz < size()) {
00159                 avail = dat + sz;
00160             }
00161         }
00162 
00163         size_type capacity() const {return limit-dat;}
00164         bool empty() const noexcept { return size() == 0;}
00165         void reserve (size_type n) {
00166             if (n > capacity()) {
00167                 grow(n);
00168             }
00169         }
00170         void shrink_to_fit(){
00171             if (limit > avail)
00172             limit = avail;
00173         }
00174 
00175     // Modifiers
00176         void clear() noexcept {
00177             uncreate();
00178         }
00179 
00180         iterator insert (const_iterator position, const value_type& val) {
00181             return insert(position, 1, val);
00182         }
00183         iterator insert(iterator position, size_type n, const value_type& val) {
00184             if (position < begin() || position > end()) {
00185                 throw std::out_of_range("Index out of range");
00186             }
00187             if (avail + n > limit) {
00188                 size_type index = position - begin();
00189                 grow(n);
00190                 position = begin() + index;
00191             }
00192             for (iterator it = end() + n - 1; it != position + n - 1; --it) {
00193                 *it = std::move(*(it - n));
```

```
00194             }
00195             std::uninitialized_fill(position, position + n, val);
00196             avail += n;
00197 
00198             return position;
00199         }
00200 
00201         iterator erase(iterator position) {
00202             if (position < dat || position > avail) {
00203                 throw std::out_of_range("Index out of range");
00204             }
00205             std::move(position + 1, avail, position);
00206             alloc.destroy(avail - 1);
00207             --avail;
00208 
00209             return position;
00210         }
00211 
00212         iterator erase(iterator first, iterator last) {
00213             iterator new_available = std::uninitialized_copy(last, avail, first);
00214 
00215             iterator it = avail;
00216             while (it != new_available) {
00217                 alloc.destroy(--it);
00218             }
00219 
00220             avail = new_available;
00221             return last;
00222         }
00223 
00224         void push_back (const value_type& t) {
00225             if (avail==limit)
00226                 grow();
00227             unchecked_append(t);
00228         }
00229 
00230         void push_back (value_type&& val) {
00231             if (avail == limit)
00232                 grow();
00233             unchecked_append(val);
00234         }
00235 
00236         void pop_back() {
00237             if (avail != dat)
00238                 alloc.destroy(--avail);
00239         }
00240 
00241         void swap(Vector& x) {
00242             std::swap(dat, x.dat);
00243             std::swap(avail, x.avail);
00244             std::swap(limit, x.limit);
00245         }
00246 
00247 
00248     // NON-MEMBER FUNCTIONS
00249 
00250         bool operator== (const Vector<T>& other) const {
00251             if (size() != other.size()) {
00252                 return false;
00253             }
00254 
00255             return std::equal(begin(), end(), other.begin());
00256         }
00257         bool operator!= (const Vector<T>& other) const {
00258             return !(*this == other);
00259         }
00260         bool operator < (const Vector<T> & other) const {
00261             return std::lexicographical_compare(begin(), end(), other.begin(), other.end());
00262         }
00263         bool operator <= (const Vector<T> & other) const {
00264             return !(other < *this);
00265         }
00266         bool operator > (const Vector<T> & other) const {
00267             return std::lexicographical_compare(other.begin(), other.end(), begin(), end());
00268         }
00269         bool operator >= (const Vector<T> & other) const {
00270             return !(other > *this);
00271         }
00272 
00273         void swap (Vector<T>& x, Vector<T>& y) {
00274             std::swap(x,y);
00275         }
00276 
00277     private:
00278         iterator dat;
00279         iterator avail;
00280         iterator limit;
```

```
00281         std::allocator<T> alloc;
00282         void create() {dat = avail = limit = nullptr;}
00283         void create (size_type n, const T& val) {
00284             dat = alloc.allocate(n);
00285             limit = avail = dat + n;
00286             std::uninitialized_fill(dat, limit, val);
00287         }
00288         void create(const_iterator i, const_iterator j) {
00289             dat = alloc.allocate(j - i);
00290             limit = avail = std::uninitialized_copy(i, j, dat);
00291         }
00292         void uncreate(){
00293             if (dat) {
00294                 iterator it = avail;
00295                 while (it != dat) {
00296                     alloc.destroy(--it);
00297                 }
00298             alloc.deallocate(dat, limit - dat);
00299             }
00300             dat = limit = avail = nullptr;
00301         }
00302         void grow(size_type new_capacity = 1) {
00303             size_type new_size = std::max(new_capacity, 2 * capacity());
00304             iterator new_data = alloc.allocate(new_size);
00305             iterator new_avail = std::uninitialized_copy(dat, avail, new_data);
00306             uncreate();
00307             dat = new_data;
00308             avail = new_avail;
00309             limit = dat + new_size;
00310         }
00311         void unchecked_append(const T& val) {
00312             alloc.construct(avail++, val);
00313         }
00314 };
```

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

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

Functions

- int main ()

Variables

- int m
- int n

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

5.8.1 Function Documentation

5.8.1.1 main()

```
int main ( )
```

5.8.2 Variable Documentation

5.8.2.1 m

```
int m
```

5.8.2.2 MAX_ND

```
int MAX_ND = 30
```

5.8.2.3 MAX_STUDENTU

```
int MAX_STUDENTU = 30
```

5.8.2.4 menu

```
int menu
```

5.8.2.5 n

```
int n
```

5.8.2.6 pasirinkimas

```
int pasirinkimas
```

5.8.2.7 raide

```
string raide
```

5.8.2.8 suma

```
double suma
```

5.8.2.9 variantas_namu_darbas

```
int variantas_namu_darbas
```

5.8.2.10 variantas_studentas

```
int variantas_studentas
```

5.8.2.11 vidurkis

```
double vidurkis
```

Index