

My Project

Generated by Doxygen 1.13.2

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Human Class Reference	7
4.1.1 Constructor & Destructor Documentation	7
4.1.1.1 Human() [1/2]	7
4.1.1.2 Human() [2/2]	8
4.1.1.3 ~Human()	8
4.1.2 Member Function Documentation	8
4.1.2.1 getPavarde()	8
4.1.2.2 getVardas()	8
4.1.2.3 setPavarde()	8
4.1.2.4 setVardas()	8
4.1.2.5 skaiciuotiGalutini()	8
4.1.3 Member Data Documentation	8
4.1.3.1 pavarde	8
4.1.3.2 vardas	9
4.2 student Class Reference	9
4.2.1 Constructor & Destructor Documentation	10
4.2.1.1 student() [1/4]	10
4.2.1.2 student() [2/4]	10
4.2.1.3 student() [3/4]	10
4.2.1.4 student() [4/4]	10
4.2.2 Member Function Documentation	11
4.2.2.1 addPazymys()	11
4.2.2.2 getEgzaminoRezultatas()	11
4.2.2.3 getGalutinisM()	11
4.2.2.4 getGalutinisV()	11
4.2.2.5 getPazymiai()	11
4.2.2.6 operator=() [1/2]	11
4.2.2.7 operator=() [2/2]	11
4.2.2.8 setEgzaminoRezultatas()	11
4.2.2.9 setGalutinisM()	11
4.2.2.10 setGalutinisV()	12
4.2.2.11 setPazymiai()	12
4.2.2.12 skaiciuotiGalutini()	12

4.2.2.13 skaiciuotiMed()	12
4.2.2.14 skaiciuotiVid()	12
4.2.3 Friends And Related Symbol Documentation	12
4.2.3.1 operator<<	12
4.2.3.2 operator>>	12
4.2.4 Member Data Documentation	12
4.2.4.1 egzaminoRezultatas	12
4.2.4.2 galutinisM	13
4.2.4.3 galutinisV	13
4.2.4.4 pazymiai	13
4.3 Vector< T > Class Template Reference	13
4.3.1 Member Typedef Documentation	14
4.3.1.1 const_iterator	14
4.3.1.2 iterator	14
4.3.2 Constructor & Destructor Documentation	15
4.3.2.1 Vector() [1/4]	15
4.3.2.2 Vector() [2/4]	15
4.3.2.3 ~Vector()	15
4.3.2.4 Vector() [3/4]	15
4.3.2.5 Vector() [4/4]	15
4.3.3 Member Function Documentation	15
4.3.3.1 assign() [1/2]	15
4.3.3.2 assign() [2/2]	15
4.3.3.3 at() [1/2]	16
4.3.3.4 at() [2/2]	16
4.3.3.5 back() [1/2]	16
4.3.3.6 back() [2/2]	16
4.3.3.7 begin() [1/2]	16
4.3.3.8 begin() [2/2]	16
4.3.3.9 capacity()	16
4.3.3.10 clear()	16
4.3.3.11 emplace_back()	17
4.3.3.12 empty()	17
4.3.3.13 end() [1/2]	17
4.3.3.14 end() [2/2]	17
4.3.3.15 erase()	17
4.3.3.16 front() [1/2]	17
4.3.3.17 front() [2/2]	17
4.3.3.18 insert()	17
4.3.3.19 operator!=(())	18
4.3.3.20 operator=() [1/3]	18
4.3.3.21 operator=() [2/3]	18

4.3.3.22 operator=() [3/3]	18
4.3.3.23 operator==()	18
4.3.3.24 operator[]() [1/2]	18
4.3.3.25 operator[]() [2/2]	18
4.3.3.26 pop_back()	18
4.3.3.27 push_back() [1/2]	19
4.3.3.28 push_back() [2/2]	19
4.3.3.29 reallocate()	19
4.3.3.30 reserve()	19
4.3.3.31 resize()	19
4.3.3.32 shrink_to_fit()	19
4.3.3.33 size()	19
4.3.3.34 swap()	19
4.3.4 Friends And Related Symbol Documentation	20
4.3.4.1 operator<	20
4.3.5 Member Data Documentation	20
4.3.5.1 capacity_	20
4.3.5.2 data	20
4.3.5.3 size_	20
5 File Documentation	21
5.1 vector/include/functions.h File Reference	21
5.1.1 Detailed Description	22
5.1.2 Function Documentation	22
5.1.2.1 antras()	22
5.1.2.2 astuntas()	22
5.1.2.3 generuotiFaila()	22
5.1.2.4 ketvirtas()	22
5.1.2.5 nuskaitytiGeneruotusFailus()	23
5.1.2.6 penktas()	23
5.1.2.7 pirmas()	23
5.1.2.8 rusiuotiOutput()	23
5.1.2.9 septintas()	23
5.1.2.10 sestas()	23
5.1.2.11 skirstytiStudentus()	24
5.1.2.12 sortedStudentSpausdinimas()	24
5.1.2.13 spausdinimas()	24
5.1.2.14 spausdinimasFaile()	24
5.1.2.15 spausdinimasTerminale()	24
5.1.2.16 trecias()	24
5.2 functions.h	25
5.3 vector/include/student.h File Reference	25

5.3.1 Detailed Description	26
5.3.2 Function Documentation	26
5.3.2.1 operator<<()	26
5.3.2.2 operator>>()	26
5.4 student.h	26
5.5 vector/include/utlis.h File Reference	28
5.5.1 Detailed Description	28
5.5.2 Function Documentation	29
5.5.2.1 tikrintInput()	29
5.6 utlis.h	29
5.7 vector/include/vector.h File Reference	29
5.7.1 Function Documentation	30
5.7.1.1 operator<()	30
5.7.1.2 operator<=()	30
5.7.1.3 operator>()	30
5.7.1.4 operator>=()	30
5.7.1.5 swap()	30
5.8 vector.h	31
5.9 vector/source/functions.cpp File Reference	34
5.9.1 Detailed Description	35
5.9.2 Function Documentation	36
5.9.2.1 antras()	36
5.9.2.2 astuntas()	36
5.9.2.3 generuotiFaila()	36
5.9.2.4 ketvirtas()	36
5.9.2.5 nuskaitytiGeneruotusFailus()	36
5.9.2.6 penktas()	36
5.9.2.7 pirmas()	36
5.9.2.8 rusiuotiOutput()	37
5.9.2.9 septintas()	37
5.9.2.10 sestas()	37
5.9.2.11 skirstytiStudentus()	37
5.9.2.12 sortedStudentSpausdinimas()	37
5.9.2.13 spausdinimas()	37
5.9.2.14 spausdinimasFaile()	38
5.9.2.15 spausdinimasTerminale()	38
5.9.2.16 trecias()	38
5.9.3 Variable Documentation	38
5.9.3.1 programosLaikas	38
5.10 vector/source/main.cpp File Reference	38
5.10.1 Detailed Description	39
5.10.2 Function Documentation	39

5.10.2.1 main()	39
5.10.3 Variable Documentation	39
5.10.3.1 programasLaikas	39
5.11 vector/source/student.cpp File Reference	39
5.11.1 Detailed Description	40
5.11.2 Function Documentation	40
5.11.2.1 operator<<()	40
5.11.2.2 operator>>()	40
Index	41

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Human	7
student	9
Vector< T >	13

Chapter 2

Class Index

2.1 Class List

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

Human	7
student	9
Vector< T >	13

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

vector/include/ functions.h	
This file contains functions declarations	21
vector/include/ student.h	
This file contains Human and Student classes	25
vector/include/ utils.h	
This file contains templates	28
vector/include/ vector.h	29
vector/source/ functions.cpp	
This file contains all functions	34
vector/source/ main.cpp	
This file handles program's flow	38
vector/source/ student.cpp	
This file contains class methods implementation	39

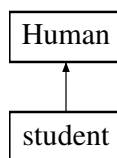
Chapter 4

Class Documentation

4.1 Human Class Reference

```
#include <student.h>
```

Inheritance diagram for Human:



Public Member Functions

- [Human](#) ()
- [Human](#) (const std::string &v, const std::string &p)
- const std::string & [getVardas](#) () const
- const std::string & [getPavarde](#) () const
- void [setVardas](#) (const std::string &v)
- void [setPavarde](#) (const std::string &p)
- virtual [~Human](#) ()=default
- virtual void [skaiciuotiGalutini](#) (char galutinioBudas)=0

Protected Attributes

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

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Human() [1/2]

```
Human::Human () [inline]
```

4.1.1.2 Human() [2/2]

```
Human::Human (  
    const std::string & v,  
    const std::string & p) [inline]
```

4.1.1.3 ~Human()

```
virtual Human::~~Human () [virtual], [default]
```

4.1.2 Member Function Documentation

4.1.2.1 getPavarde()

```
const std::string & Human::getPavarde () const [inline]
```

4.1.2.2 getVardas()

```
const std::string & Human::getVardas () const [inline]
```

4.1.2.3 setPavarde()

```
void Human::setPavarde (  
    const std::string & p) [inline]
```

4.1.2.4 setVardas()

```
void Human::setVardas (  
    const std::string & v) [inline]
```

4.1.2.5 skaiciuotiGalutini()

```
virtual void Human::skaiciuotiGalutini (  
    char galutinioBudas) [pure virtual]
```

Implemented in [student](#).

4.1.3 Member Data Documentation

4.1.3.1 pavarde

```
std::string Human::pavarde [protected]
```


4.1.3.2 vardas

```
std::string Human::vardas [protected]
```

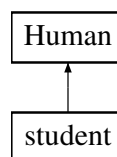
The documentation for this class was generated from the following file:

- vector/include/[student.h](#)

4.2 student Class Reference

```
#include <student.h>
```

Inheritance diagram for student:



Public Member Functions

- [student](#) ()=default
- [student](#) (const [student](#) &other) noexcept
- [student](#) & [operator=](#) (const [student](#) &other) noexcept
- [student](#) ([student](#) &&other) noexcept
- [student](#) & [operator=](#) ([student](#) &&other) noexcept
- [student](#) (std::string v, std::string p, [Vector](#)< float > [pazymiai](#), float egz) noexcept
- void [setPazymiai](#) ([Vector](#)< float > paz) noexcept
- void [setEgzaminoRezultatas](#) (float egz) noexcept
- void [setGalutinisV](#) (float V) noexcept
- void [setGalutinisM](#) (float M) noexcept
- const [Vector](#)< float > & [getPazymiai](#) () const
- float [getEgzaminoRezultatas](#) () const
- float [getGalutinisV](#) () const
- float [getGalutinisM](#) () const
- float [skaiciuotiVid](#) () const
- float [skaiciuotiMed](#) () const
- void [skaiciuotiGalutini](#) (char galutinioBudas) override
- void [addPazymys](#) (float pazymys)

Public Member Functions inherited from [Human](#)

- [Human](#) ()
- [Human](#) (const std::string &v, const std::string &p)
- const std::string & [getVardas](#) () const
- const std::string & [getPavarde](#) () const
- void [setVardas](#) (const std::string &v)
- void [setPavarde](#) (const std::string &p)
- virtual [~Human](#) ()=default

Private Attributes

- `Vector< float > pazymiai {}`
- `float egzaminoRezultatas = 0.0f`
- `float galutinisM = 0.0f`
- `float galutinisV = 0.0f`

Friends

- `std::ostream & operator<< (std::ostream &os, const student &studentas)`
- `std::istream & operator>> (std::istream &in, student &studentas)`

Additional Inherited Members

Protected Attributes inherited from `Human`

- `std::string vardas`
- `std::string pavarde`

4.2.1 Constructor & Destructor Documentation

4.2.1.1 `student()` [1/4]

```
student::student () [default]
```

4.2.1.2 `student()` [2/4]

```
student::student (  
    const student & other) [inline], [noexcept]
```

4.2.1.3 `student()` [3/4]

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

4.2.1.4 `student()` [4/4]

```
student::student (  
    std::string v,  
    std::string p,  
    Vector< float > pazymiai,  
    float egz) [inline], [noexcept]
```

4.2.2 Member Function Documentation

4.2.2.1 addPazymys()

```
void student::addPazymys (  
    float pazymys) [inline]
```

4.2.2.2 getEgzaminoRezultatas()

```
float student::getEgzaminoRezultatas () const [inline]
```

4.2.2.3 getGalutinisM()

```
float student::getGalutinisM () const [inline]
```

4.2.2.4 getGalutinisV()

```
float student::getGalutinisV () const [inline]
```

4.2.2.5 getPazymiai()

```
const Vector< float > & student::getPazymiai () const [inline]
```

4.2.2.6 operator=() [1/2]

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

4.2.2.7 operator=() [2/2]

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

4.2.2.8 setEgzaminoRezultatas()

```
void student::setEgzaminoRezultatas (  
    float egz) [inline], [noexcept]
```

4.2.2.9 setGalutinisM()

```
void student::setGalutinisM (  
    float M) [inline], [noexcept]
```

4.2.2.10 setGalutinisV()

```
void student::setGalutinisV (  
    float V) [inline], [noexcept]
```

4.2.2.11 setPazymiai()

```
void student::setPazymiai (  
    Vector< float > paz) [inline], [noexcept]
```

4.2.2.12 skaiciuotiGalutini()

```
void student::skaiciuotiGalutini (  
    char galutinioBudas) [inline], [override], [virtual]
```

Implements [Human](#).

4.2.2.13 skaiciuotiMed()

```
float student::skaiciuotiMed () const
```

4.2.2.14 skaiciuotiVid()

```
float student::skaiciuotiVid () const
```

4.2.3 Friends And Related Symbol Documentation

4.2.3.1 operator<<

```
std::ostream & operator<< (  
    std::ostream & os,  
    const student & studentas) [friend]
```

4.2.3.2 operator>>

```
std::istream & operator>> (  
    std::istream & in,  
    student & studentas) [friend]
```

4.2.4 Member Data Documentation

4.2.4.1 egzaminoRezultatas

```
float student::egzaminoRezultatas = 0.0f [private]
```

4.2.4.2 galutinisM

```
float student::galutinisM = 0.0f [private]
```

4.2.4.3 galutinisV

```
float student::galutinisV = 0.0f [private]
```

4.2.4.4 pazymiai

```
Vector<float> student::pazymiai {} [private]
```

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

- vector/include/[student.h](#)
- vector/source/[student.cpp](#)

4.3 Vector< T > Class Template Reference

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

Public Types

- using [iterator](#) = T*
- using [const_iterator](#) = const T*

Public Member Functions

- [Vector](#) ()
- [Vector](#) (std::initializer_list< T > init)
- [~Vector](#) ()
- [Vector](#) (const [Vector](#) &other)
- [Vector](#) & [operator=](#) (const [Vector](#) &other)
- [Vector](#) ([Vector](#) &&other) noexcept
- [Vector](#) & [operator=](#) ([Vector](#) &&other) noexcept
- bool [operator==](#) (const [Vector](#) &o) const
- bool [operator!=](#) (const [Vector](#) &o) const
- size_t [size](#) () const
- size_t [capacity](#) () const
- bool [empty](#) () const
- void [reserve](#) (size_t newCap)
- void [shrink_to_fit](#) ()
- void [clear](#) () noexcept
- void [resize](#) (size_t count, const T &value=T())
- void [push_back](#) (const T &value)
- void [push_back](#) (T &&value)
- void [pop_back](#) ()

- void `swap` (`Vector` &other) noexcept
- void `assign` (size_t n, const T &value)
- template<typename InputIt>
void `assign` (InputIt first, InputIt last)
- `iterator begin` ()
- `const_iterator begin` () const
- `iterator end` ()
- `const_iterator end` () const
- `iterator insert` (`const_iterator` pos, const T &value)
- `iterator erase` (`const_iterator` pos)
- template<typename... Args>
void `emplace_back` (Args &&... args)
- T & `operator[]` (size_t i)
- const T & `operator[]` (size_t i) const
- T & `at` (size_t i)
- const T & `at` (size_t i) const
- T & `front` ()
- const T & `front` () const
- T & `back` ()
- const T & `back` () const
- `Vector` & `operator=` (std::initializer_list< T > init)

Private Member Functions

- void `reallocate` (size_t newCap)

Private Attributes

- T * `data`
- size_t `size_`
- size_t `capacity_`

Friends

- template<typename U>
bool `operator<` (const `Vector`< U > &a, const `Vector`< U > &b)

4.3.1 Member Typedef Documentation

4.3.1.1 `const_iterator`

```
template<typename T>
using Vector< T >::const_iterator = const T*
```

4.3.1.2 `iterator`

```
template<typename T>
using Vector< T >::iterator = T*
```

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Vector() [1/4]

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

4.3.2.2 Vector() [2/4]

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

4.3.2.3 ~Vector()

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

4.3.2.4 Vector() [3/4]

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

4.3.2.5 Vector() [4/4]

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

4.3.3 Member Function Documentation

4.3.3.1 assign() [1/2]

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

4.3.3.2 assign() [2/2]

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

4.3.3.3 at() [1/2]

```
template<typename T>
T & Vector< T >::at (
    size_t i) [inline]
```

4.3.3.4 at() [2/2]

```
template<typename T>
const T & Vector< T >::at (
    size_t i) const [inline]
```

4.3.3.5 back() [1/2]

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

4.3.3.6 back() [2/2]

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

4.3.3.7 begin() [1/2]

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

4.3.3.8 begin() [2/2]

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

4.3.3.9 capacity()

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

4.3.3.10 clear()

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


4.3.3.11 `emplace_back()`

```
template<typename T>
template<typename... Args>
void Vector< T >::emplace_back (
    Args &&... args) [inline]
```

4.3.3.12 `empty()`

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

4.3.3.13 `end()` [1/2]

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

4.3.3.14 `end()` [2/2]

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

4.3.3.15 `erase()`

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

4.3.3.16 `front()` [1/2]

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

4.3.3.17 `front()` [2/2]

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

4.3.3.18 `insert()`

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

4.3.3.19 operator!=(())

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

4.3.3.20 operator=() [1/3]

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

4.3.3.21 operator=() [2/3]

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

4.3.3.22 operator=() [3/3]

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

4.3.3.23 operator==(())

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

4.3.3.24 operator[]() [1/2]

```
template<typename T>
T & Vector< T >::operator[] (
    size_t i) [inline]
```

4.3.3.25 operator[]() [2/2]

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

4.3.3.26 pop_back()

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

4.3.3.27 push_back() [1/2]

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

4.3.3.28 push_back() [2/2]

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

4.3.3.29 reallocate()

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

4.3.3.30 reserve()

```
template<typename T>
void Vector< T >::reserve (
    size_t newCap) [inline]
```

4.3.3.31 resize()

```
template<typename T>
void Vector< T >::resize (
    size_t count,
    const T & value = T()) [inline]
```

4.3.3.32 shrink_to_fit()

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

4.3.3.33 size()

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

4.3.3.34 swap()

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

4.3.4 Friends And Related Symbol Documentation

4.3.4.1 operator<

```
template<typename T>
template<typename U>
bool operator< (
    const Vector< U > & a,
    const Vector< U > & b) [friend]
```

4.3.5 Member Data Documentation

4.3.5.1 capacity_

```
template<typename T>
size_t Vector< T >::capacity_ [private]
```

4.3.5.2 data

```
template<typename T>
T* Vector< T >::data [private]
```

4.3.5.3 size_

```
template<typename T>
size_t Vector< T >::size_ [private]
```

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

- vector/include/[vector.h](#)

Chapter 5

File Documentation

5.1 vector/include/functions.h File Reference

this file contains functions declarations

```
#include <vector>
#include "../include/student.h"
#include "../include/vector.h"
```

Functions

- void [rusiuotiOutput](#) (Vector< [student](#) > &grupe, char rusiavimoBudas, char galutinioBudas)
- void [spausdinimasTerminale](#) (const Vector< [student](#) > &grupe, char galutinioBudas)
- void [spausdinimasFaile](#) (const Vector< [student](#) > &grupe, char galutinioBudas)
- void [generuotiFaile](#) (int pKiekis, int studentuKiekis, const std::string &failoPavadinimas)
- void [spausdinimas](#) (char spausBudas, char rusiavimoBudas, char galutinioBudas, Vector< [student](#) > &grupe)
- void [sortedStudentSpausdinimas](#) (std::string lowGradeFailas, std::string highGradeFailas, Vector< [student](#) > &nepazangus, Vector< [student](#) > &normalus, char galutinioBudas)
- void [nuskaitytiGeneruotusFailus](#) (const std::string &failoPavadinimas, Vector< [student](#) > &grupe, int pKiekis, char galutinioBudas)
- void [skirstytiStudentus](#) (Vector< [student](#) > &grupe, Vector< [student](#) > &nepazangus, Vector< [student](#) > &normalus, char galutinioBudas)
- void [pirmas](#) (Vector< [student](#) > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void [antras](#) (Vector< [student](#) > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void [trecias](#) (Vector< [student](#) > &grupe, const Vector< std::string > &vardai, const Vector< std::string > &pavardes, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void [ketvirtas](#) (Vector< [student](#) > &grupe, int pKiekis, char galutinioBudas)
- void [penktas](#) (int pKiekis)
- void [sestas](#) (Vector< [student](#) > &grupe, Vector< [student](#) > &testGrupe, Vector< [student](#) > &nepazangus, Vector< [student](#) > &normalus, char galutinioBudas, char rusiavimoBudas, int pKiekis)
- void [septintas](#) (char galutinioBudas)
- void [astuntas](#) ()

5.1.1 Detailed Description

this file contains functions declarations

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.1.2 Function Documentation

5.1.2.1 antras()

```
void antras (  
    Vector< student > & grupe,  
    char spausBudas,  
    char rusiavimoBudas,  
    char galutinioBudas,  
    int pKiekis)
```

5.1.2.2 astuntas()

```
void astuntas ()
```

5.1.2.3 generuotiFaila()

```
void generuotiFaila (  
    int pKiekis,  
    int studentuKiekis,  
    const std::string & failoPavadinimas)
```

5.1.2.4 ketvirtas()

```
void ketvirtas (  
    Vector< student > & grupe,  
    int pKiekis,  
    char galutinioBudas)
```

5.1.2.5 nuskaitytiGeneruotusFailus()

```
void nuskaitytiGeneruotusFailus (
    const std::string & failoPavadinimas,
    Vector< student > & grupe,
    int pKiekis,
    char galutinioBudas)
```

5.1.2.6 penktas()

```
void penktas (
    int pKiekis)
```

5.1.2.7 pirmas()

```
void pirmas (
    Vector< student > & grupe,
    char spausBudas,
    char rusiavimoBudas,
    char galutinioBudas,
    int pKiekis)
```

5.1.2.8 rusiuotiOutput()

```
void rusiuotiOutput (
    Vector< student > & grupe,
    char rusiavimoBudas,
    char galutinioBudas)
```

5.1.2.9 septintas()

```
void septintas (
    char galutinioBudas)
```

5.1.2.10 sestasis()

```
void sestasis (
    Vector< student > & grupe,
    Vector< student > & testGrupe,
    Vector< student > & nepazangus,
    Vector< student > & normalus,
    char galutinioBudas,
    char rusiavimoBudas,
    int pKiekis)
```

5.1.2.11 skirstytiStudentus()

```
void skirstytiStudentus (
    Vector< student > & grupe,
    Vector< student > & nepazangus,
    Vector< student > & normalus,
    char galutinioBudas)
```

5.1.2.12 sortedStudentSpausdinimas()

```
void sortedStudentSpausdinimas (
    std::string lowGradeFailas,
    std::string highGradeFailas,
    Vector< student > & nepazangus,
    Vector< student > & normalus,
    char galutinioBudas)
```

5.1.2.13 spausdinimas()

```
void spausdinimas (
    char spausBudas,
    char rusiavimoBudas,
    char galutinioBudas,
    Vector< student > & grupe)
```

5.1.2.14 spausdinimasFaile()

```
void spausdinimasFaile (
    const Vector< student > & grupe,
    char galutinioBudas)
```

5.1.2.15 spausdinimasTerminale()

```
void spausdinimasTerminale (
    const Vector< student > & grupe,
    char galutinioBudas)
```

5.1.2.16 trecias()

```
void trecias (
    Vector< student > & grupe,
    const Vector< std::string > & vardai,
    const Vector< std::string > & pavardes,
    char spausBudas,
    char rusiavimoBudas,
    char galutinioBudas,
    int pKiekis)
```


5.2 functions.h

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

```
00001
00011
00012 #ifndef FUNCTIONS_H
00013 #define FUNCTIONS_H
00014
00015 #include <vector>
00016 #include "../include/student.h"
00017 #include "../include/vector.h"
00018
00019 void rusiuotiOutput(Vector<student>& grupe, char rusiavimoBudas, char galutinioBudas);
00020 void spausdinimasTerminale(const Vector<student>& grupe, char galutinioBudas);
00021 void spausdinimasFaile(const Vector<student>& grupe, char galutinioBudas);
00022 void generuotiFaila(int pKiekis, int studentuKiekis, const std::string& failoPavadinimas);
00023 void spausdinimas(char spausBudas, char rusiavimoBudas, char galutinioBudas, Vector<student>&grupe);
00024 void sortedStudentSpausdinimas(std::string lowGradeFailas, std::string highGradeFailas,
    Vector<student>&nepazangus, Vector<student>&normalus, char galutinioBudas);
00025 void nuskaitytiGeneruotusFailus(const std::string& failoPavadinimas, Vector<student> & grupe, int
    pKiekis, char galutinioBudas);
00026 void skirstytiStudentus(Vector<student> & grupe, Vector<student>& nepazangus, Vector<student>&
    normalus, char galutinioBudas);
00027
00028 void pirmas(Vector<student>& grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int
    pKiekis);
00029 void antras(Vector<student>& grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int
    pKiekis);
00030 void trecias(Vector<student>& grupe, const Vector<std::string>& vardai, const Vector<std::string>&
    pavardes, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis);
00031 void ketvirtas(Vector<student>& grupe, int pKiekis, char galutinioBudas);
00032 void penktas(int pKiekis);
00033 void sestas(Vector<student>& grupe, Vector<student>& testGrupe, Vector<student>& nepazangus,
    Vector<student>& normalus, char galutinioBudas, char rusiavimoBudas, int pKiekis);
00034 void septintas(char galutinioBudas);
00035 void astuntas();
00036 #endif
```

5.3 vector/include/student.h File Reference

this file contains [Human](#) and Student classes

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

Classes

- class [Human](#)
- class [student](#)

Functions

- std::ostream & [operator<<](#) (std::ostream &os, const [student](#) &studentas)
- std::istream & [operator>>](#) (std::istream &in, [student](#) &studentas)

5.3.1 Detailed Description

this file contains [Human](#) and Student classes

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.3.2 Function Documentation

5.3.2.1 operator<<()

```
std::ostream & operator<< (  
    std::ostream & os,  
    const student & studentas)
```

5.3.2.2 operator>>()

```
std::istream & operator>> (  
    std::istream & in,  
    student & studentas)
```

5.4 student.h

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

```
00001 #ifndef STUDENT_H  
00002 #define STUDENT_H  
00003  
00014  
00015 #include <vector>  
00016 #include <string>  
00017 #include <iostream>  
00018 #include "../include/vector.h"  
00019 class Human{  
00020     protected:  
00021  
00022         std::string vardas;  
00023         std::string pavarde;  
00024  
00025     public:  
00026  
00027     Human() : vardas(""), pavarde("") {}  
00028     Human(const std::string& v, const std::string& p) : vardas(v), pavarde(p) {}
```

```

00029
00030 //getters
00031 const std::string& getVardas() const { return vardas; }
00032 const std::string& getPavarde() const { return pavarde; }
00033 //setters
00034 void setVardas(const std::string& v) { vardas = v; }
00035 void setPavarde(const std::string& p) { pavarde = p; }
00036
00037 virtual ~Human() = default;
00038
00039 virtual void skaiciuotiGalutini(char galutinioBudas) = 0;
00040
00041 };
00042
00043 class student : public Human{
00044
00045     private:
00046
00047         Vector<float> pazymiai{};
00048         float egzaminoRezultatas = 0.0f;
00049         float galutinisM = 0.0f;
00050         float galutinisV = 0.0f;
00051
00052     public:
00053
00054         student() = default;
00055
00056         //rule of 5-----
00057         //copy
00058         student(const student &other) noexcept : Human(other.getVardas(), other.getPavarde()),
00059         pazymiai(other.pazymiai),
00060         egzaminoRezultatas(other.egzaminoRezultatas),
00061         galutinisM(other.galutinisM), galutinisV(other.galutinisV){};
00062         //copy asg
00063         student& operator=(const student &other) noexcept {
00064             if(this != &other){
00065                 setVardas(other.getVardas());
00066                 setPavarde(other.getPavarde());
00067                 pazymiai = other.pazymiai;
00068                 egzaminoRezultatas = other.egzaminoRezultatas;
00069                 galutinisM = other.galutinisM;
00070                 galutinisV = other.galutinisV;
00071             }
00072             return *this;
00073         };
00074         //move
00075         student(student &&other) noexcept : Human(std::move(other.vardas), std::move(other.pavarde)),
00076         pazymiai(std::move(other.pazymiai)),
00077         egzaminoRezultatas(other.egzaminoRezultatas),
00078         galutinisM(other.galutinisM),
00079         galutinisV(other.galutinisV) {
00080             other.setEgzaminoRezultatas(0);
00081             other.setGalutinisM(0);
00082             other.setGalutinisV(0);
00083         }
00084         //move asg
00085         student& operator=(student &&other) noexcept {
00086             if(this != &other){
00087                 vardas = std::move(other.vardas);
00088                 pavarde = std::move(other.pavarde);
00089                 pazymiai = std::move(other.pazymiai);
00090                 egzaminoRezultatas = other.egzaminoRezultatas;
00091                 galutinisM = other.galutinisM;
00092                 galutinisV = other.galutinisV;
00093                 other.setEgzaminoRezultatas(0);
00094                 other.setGalutinisM(0);
00095                 other.setGalutinisV(0);
00096             }
00097             return *this;
00098         };
00099         // -----
00100
00101         //isvestis, ivestis overloads
00102         friend std::ostream& operator << (std::ostream& os, const student& studentas);
00103         friend std::istream& operator >> (std::istream& in, student& studentas);
00104
00105         //parametrizuotas ctor
00106         student(std::string v, std::string p, Vector<float> pazymiai, float egz) noexcept : Human(v, p),
00107         pazymiai(std::move(pazymiai)), egzaminoRezultatas(egz) {}
00108
00109         //setters
00110         void setPazymiai(Vector<float> paz) noexcept { pazymiai = std::move(paz); }
00111         void setEgzaminoRezultatas(float egz) noexcept { egzaminoRezultatas = egz; }
00112         void setGalutinisV(float V) noexcept { galutinisV = V; }
00113         void setGalutinisM(float M) noexcept { galutinisM = M; }

```

```

00113
00114 //getters
00115 const Vector<float>& getPazymiai() const { return pazymiai; }
00116 float getEgzaminoRezultatas() const { return egzaminoRezultatas; }
00117 float getGalutinisV() const { return galutinisV; }
00118 float getGalutinisM() const { return galutinisM; }
00119
00120 //methods
00121 float skaiciuotiVid() const;
00122 float skaiciuotiMed() const;
00123 void skaiciuotiGalutini(char galutinioBudat) override;
00124 void addPazymys(float pazymys) { pazymiai.push_back(pazymys); }
00125 };
00126
00127 std::ostream& operator<<(std::ostream& os, const student& studentas);
00128 std::istream& operator>>(std::istream& in, student& studentas);
00129
00130 #endif

```

5.5 vector/include/utlis.h File Reference

this file contains templates

```

#include <iostream>
#include <limits>
#include <string>

```

Functions

- template<typename T>
T [tikrintiInput](#) (const std::string &prompt, const std::string &klaida)

5.5.1 Detailed Description

this file contains templates

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.5.2 Function Documentation

5.5.2.1 tikrintiInput()

```
template<typename T>
T tikrintiInput (
    const std::string & prompt,
    const std::string & klaida)
```

5.6 utils.h

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

```
00001 #ifndef UTILS_H
00002 #define UTILS_H
00003
00014
00015 #include <iostream>
00016 #include <limits>
00017 #include <string>
00018
00019 template<typename T>
00020 T tikrintiInput(const std::string& prompt, const std::string& klaida) {
00021     T value;
00022     while (true) {
00023         std::cout << prompt;
00024         std::cin >> value;
00025         if (!std::cin.fail()) {
00026             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00027             return value;
00028         }
00029         std::cerr << klaida << std::endl;
00030         std::cin.clear();
00031         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00032     }
00033 }
00034
00035 #endif
```

5.7 vector/include/vector.h File Reference

```
#include <iostream>
#include <initializer_list>
#include <stdexcept>
#include <algorithm>
#include <iterator>
#include <utility>
#include <memory>
```

Classes

- class [Vector< T >](#)

Functions

- `template<typename T>`
`void swap (Vector< T > &a, Vector< T > &b) noexcept`
- `template<typename T>`
`bool operator< (const Vector< T > &a, const Vector< T > &b)`
- `template<typename T>`
`bool operator> (const Vector< T > &a, const Vector< T > &b)`
- `template<typename T>`
`bool operator<= (const Vector< T > &a, const Vector< T > &b)`
- `template<typename T>`
`bool operator>= (const Vector< T > &a, const Vector< T > &b)`

5.7.1 Function Documentation

5.7.1.1 operator<()

```
template<typename T>
bool operator< (
    const Vector< T > & a,
    const Vector< T > & b)
```

5.7.1.2 operator<=()

```
template<typename T>
bool operator<= (
    const Vector< T > & a,
    const Vector< T > & b)
```

5.7.1.3 operator>()

```
template<typename T>
bool operator> (
    const Vector< T > & a,
    const Vector< T > & b)
```

5.7.1.4 operator>=()

```
template<typename T>
bool operator>= (
    const Vector< T > & a,
    const Vector< T > & b)
```

5.7.1.5 swap()

```
template<typename T>
void swap (
    Vector< T > & a,
    Vector< T > & b) [noexcept]
```

5.8 vector.h

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

```

00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003
00004 #include <iostream>
00005 #include <initializer_list>
00006 #include <stdexcept>
00007 #include <algorithm>
00008 #include <iterator>
00009 #include <utility>
00010 #include <memory>
00011
00012 template<typename T>
00013 class Vector {
00014 private:
00015     T* data;
00016     size_t size_;
00017     size_t capacity_;
00018
00019     void reallocate(size_t newCap) {
00020         T* newData = static_cast<T*> (::operator new(newCap * sizeof(T)));
00021         for (size_t i = 0; i < size_; ++i) {
00022             try {
00023                 new(newData + i) T(std::move(data[i]));
00024             } catch (...) {
00025                 for (size_t j = 0; j < i; ++j) {
00026                     (newData + j)->~T();
00027                 }
00028                 ::operator delete(newData);
00029                 throw;
00030             }
00031         }
00032         for (size_t i = 0; i < size_; ++i) {
00033             (data + i)->~T();
00034         }
00035         ::operator delete(data);
00036         data = newData;
00037         capacity_ = newCap;
00038     }
00039
00040 public:
00041     // default ctor
00042     Vector() : data(nullptr), size_(0), capacity_(0) {}
00043
00044     // initializer-list ctor
00045     Vector(std::initializer_list<T> init)
00046         : data(nullptr), size_(0), capacity_(0)
00047     {
00048         reserve(init.size());
00049         for (const auto& x : init) {
00050             push_back(x);
00051         }
00052     }
00053
00054     ~Vector() {
00055         clear();
00056         ::operator delete(data);
00057     }
00058
00059     Vector(const Vector& other)
00060         : data(nullptr), size_(0), capacity_(0)
00061     {
00062         if (other.size_ > 0) {
00063             reserve(other.capacity_);
00064             for (size_t i = 0; i < other.size_; ++i) {
00065                 try {
00066                     new(data + i) T(other.data[i]);
00067                     ++size_;
00068                 } catch (...) {
00069                     for (size_t j = 0; j < i; ++j) {
00070                         (data + j)->~T();
00071                     }
00072                     ::operator delete(data);
00073                     data = nullptr;
00074                     size_ = 0;
00075                     capacity_ = 0;
00076                     throw;
00077                 }
00078             }
00079         }
00080     }
00081 }
00082

```

```

00083     Vector& operator=(const Vector& other) {
00084         if (this != &other) {
00085             Vector temp(other);
00086             swap(temp);
00087         }
00088         return *this;
00089     }
00090
00091     Vector(Vector&& other) noexcept
00092     : data(other.data), size_(other.size_), capacity_(other.capacity_)
00093     {
00094         other.data = nullptr;
00095         other.size_ = 0;
00096         other.capacity_ = 0;
00097     }
00098
00099     Vector& operator=(Vector&& other) noexcept {
00100         if (this != &other) {
00101             clear();
00102             ::operator delete(data);
00103
00104             data = other.data;
00105             size_ = other.size_;
00106             capacity_ = other.capacity_;
00107
00108             other.data = nullptr;
00109             other.size_ = 0;
00110             other.capacity_ = 0;
00111         }
00112         return *this;
00113     }
00114
00115     bool operator==(const Vector& o) const {
00116         if (size_ != o.size_) return false;
00117         for (size_t i = 0; i < size_; ++i)
00118             if (data[i] != o.data[i]) return false;
00119         return true;
00120     }
00121
00122     bool operator!=(const Vector& o) const {
00123         return !(*this == o);
00124     }
00125
00126     size_t size() const { return size_; }
00127     size_t capacity() const { return capacity_; }
00128     bool empty() const { return size_ == 0; }
00129
00130     void reserve(size_t newCap) {
00131         if (newCap > capacity_) {
00132             if (capacity_ == 0) {
00133                 data = static_cast<T*> (::operator new(newCap * sizeof(T)));
00134                 capacity_ = newCap;
00135             } else {
00136                 reallocate(newCap);
00137             }
00138         }
00139     }
00140
00141     void shrink_to_fit() {
00142         if (capacity_ > size_) {
00143             if (size_ == 0) {
00144                 ::operator delete(data);
00145                 data = nullptr;
00146                 capacity_ = 0;
00147             } else {
00148                 reallocate(size_);
00149             }
00150         }
00151     }
00152
00153     void clear() noexcept {
00154         for (size_t i = 0; i < size_; ++i) {
00155             (data + i)->~T();
00156         }
00157         size_ = 0;
00158     }
00159
00160     void resize(size_t count, const T& value = T()) {
00161         if (count > capacity_) {
00162             reserve(count);
00163         }
00164
00165         if (count > size_) {
00166             // Construct new elements
00167             for (size_t i = size_; i < count; ++i) {
00168                 new(data + i) T(value);
00169             }

```



```

00170         } else if (count < size_) {
00171             // Destroy excess elements
00172             for (size_t i = count; i < size_; ++i) {
00173                 (data + i)->~T();
00174             }
00175         }
00176         size_ = count;
00177     }
00178
00179     void push_back(const T& value) {
00180         if (size_ == capacity_) {
00181             reserve(capacity_ == 0 ? 1 : capacity_ * 2);
00182         }
00183         new(data + size_) T(value);
00184         ++size_;
00185     }
00186
00187     void push_back(T&& value) {
00188         if (size_ == capacity_) {
00189             reserve(capacity_ == 0 ? 1 : capacity_ * 2);
00190         }
00191         new(data + size_) T(std::move(value));
00192         ++size_;
00193     }
00194
00195     void pop_back() {
00196         if (size_ == 0) {
00197             throw std::out_of_range("pop_back() on empty Vector");
00198         }
00199         --size_;
00200         (data + size_)->~T();
00201     }
00202
00203     void swap(Vector& other) noexcept {
00204         std::swap(data, other.data);
00205         std::swap(size_, other.size_);
00206         std::swap(capacity_, other.capacity_);
00207     }
00208
00209     void assign(size_t n, const T& value) {
00210         clear();
00211         if (n > capacity_) {
00212             reserve(n);
00213         }
00214         for (size_t i = 0; i < n; ++i) {
00215             new(data + i) T(value);
00216         }
00217         size_ = n;
00218     }
00219
00220     template<typename InputIt>
00221     void assign(InputIt first, InputIt last) {
00222         clear();
00223         for (; first != last; ++first) {
00224             push_back(*first);
00225         }
00226     }
00227
00228     using iterator = T*;
00229     using const_iterator = const T*;
00230
00231     iterator begin() { return data; }
00232     const_iterator begin() const { return data; }
00233     iterator end() { return data + size_; }
00234     const_iterator end() const { return data + size_; }
00235
00236     iterator insert(const_iterator pos, const T& value) {
00237         size_t idx = pos - data;
00238         if (size_ == capacity_) {
00239             reserve(capacity_ == 0 ? 1 : capacity_ * 2);
00240         }
00241
00242         for (size_t i = size_; i > idx; --i) {
00243             new(data + i) T(std::move(data[i-1]));
00244             (data + i - 1)->~T();
00245         }
00246
00247         new(data + idx) T(value);
00248         ++size_;
00249         return data + idx;
00250     }
00251
00252     iterator erase(const_iterator pos) {
00253         size_t idx = pos - data;
00254         if (idx >= size_) {
00255             return end();
00256         }

```

```

00257         (data + idx)->~T();
00258     }
00259     for (size_t i = idx; i + 1 < size_; ++i) {
00260         new(data + i) T(std::move(data[i + 1]));
00261         (data + i + 1)->~T();
00262     }
00263     }
00264     --size_;
00265     return data + idx;
00266 }
00267
00268 template<typename... Args>
00269 void emplace_back(Args&&... args) {
00270     if (size_ == capacity_) {
00271         reserve(capacity_ == 0 ? 1 : capacity_ * 2);
00272     }
00273     new(data + size_) T(std::forward<Args>(args)...);
00274     ++size_;
00275 }
00276
00277 T& operator[](size_t i) { return data[i]; }
00278 const T& operator[](size_t i) const { return data[i]; }
00279
00280 T& at(size_t i) {
00281     if (i >= size_) throw std::out_of_range("Vector::at");
00282     return data[i];
00283 }
00284 const T& at(size_t i) const {
00285     if (i >= size_) throw std::out_of_range("Vector::at");
00286     return data[i];
00287 }
00288
00289 T& front() { return at(0); }
00290 const T& front() const { return at(0); }
00291 T& back() { return at(size_-1); }
00292 const T& back() const { return at(size_-1); }
00293
00294 // assign from initializer_list
00295 Vector& operator=(std::initializer_list<T> init) {
00296     assign(init.begin(), init.end());
00297     return *this;
00298 }
00299
00300 template<typename U>
00301 friend bool operator< (const Vector<U> &a, const Vector<U> &b);
00302 };
00303
00304 template<typename T>
00305 void swap(Vector<T> &a, Vector<T> &b) noexcept {
00306     a.swap(b);
00307 }
00308
00309 template<typename T>
00310 bool operator< (const Vector<T> &a, const Vector<T> &b) {
00311     return std::lexicographical_compare(
00312         a.begin(), a.end(),
00313         b.begin(), b.end()
00314     );
00315 }
00316
00317 template<typename T>
00318 bool operator> (const Vector<T> &a, const Vector<T> &b) { return b < a; }
00319
00320 template<typename T>
00321 bool operator<= (const Vector<T> &a, const Vector<T> &b) { return !(b < a); }
00322
00323 template<typename T>
00324 bool operator>= (const Vector<T> &a, const Vector<T> &b) { return !(a < b); }
00325
00326 #endif // VECTOR_H

```

5.9 vector/source/functions.cpp File Reference

this file contains all functions

```

#include "functions.h"
#include <numeric>
#include <algorithm>

```

```
#include <iomanip>
#include <fstream>
#include <iostream>
#include <chrono>
#include <sstream>
#include <random>
#include "student.h"
#include "vector.h"
```

Functions

- void [rusiuotiOutput](#) ([Vector](#)< [student](#) > &grupe, char rusiavimoBudas, char galutinioBudas)
- void [spausdinimasTerminale](#) (const [Vector](#)< [student](#) > &grupe, char galutinioBudas)
- void [spausdinimasFaile](#) (const [Vector](#)< [student](#) > &grupe, char galutinioBudas)
- void [generuotiFaila](#) (int pKiekis, int studentuKiekis, const std::string &failoPavadinimas)
- void [spausdinimas](#) (char spausBudas, char rusiavimoBudas, char galutinioBudas, [Vector](#)< [student](#) > &grupe)
- void [sortedStudentSpausdinimas](#) (std::string lowGradeFailas, std::string highGradeFailas, [Vector](#)< [student](#) > &nepazangus, [Vector](#)< [student](#) > &normalus, char galutinioBudas)
- void [nuskaitytiGeneruotusFailus](#) (const std::string &failoPavadinimas, [Vector](#)< [student](#) > &grupe, int pKiekis, char galutinioBudas)
- void [skirstytiStudentus](#) ([Vector](#)< [student](#) > &grupe, [Vector](#)< [student](#) > &nepazangus, [Vector](#)< [student](#) > &normalus, char galutinioBudas)
- void [pirmas](#) ([Vector](#)< [student](#) > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void [antras](#) ([Vector](#)< [student](#) > &grupe, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void [trecias](#) ([Vector](#)< [student](#) > &grupe, const [Vector](#)< std::string > &vardai, const [Vector](#)< std::string > &pavardes, char spausBudas, char rusiavimoBudas, char galutinioBudas, int pKiekis)
- void [ketvirtas](#) ([Vector](#)< [student](#) > &grupe, int pKiekis, char galutinioBudas)
- void [penktas](#) (int pKiekis)
- void [sestas](#) ([Vector](#)< [student](#) > &grupe, [Vector](#)< [student](#) > &testGrupe, [Vector](#)< [student](#) > &nepazangus, [Vector](#)< [student](#) > &normalus, char galutinioBudas, char rusiavimoBudas, int pKiekis)
- void [septintas](#) (char galutinioBudas)
- void [astuntas](#) ()

Variables

- double [programosLaikas](#)

5.9.1 Detailed Description

this file contains all functions

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.9.2 Function Documentation

5.9.2.1 antras()

```
void antras (  
    Vector< student > & grupe,  
    char spausBudas,  
    char rusiavimoBudas,  
    char galutinioBudas,  
    int pKiekis)
```

5.9.2.2 astuntas()

```
void astuntas ()
```

5.9.2.3 generuotiFaila()

```
void generuotiFaila (  
    int pKiekis,  
    int studentuKiekis,  
    const std::string & failoPavadinimas)
```

5.9.2.4 ketvirtas()

```
void ketvirtas (  
    Vector< student > & grupe,  
    int pKiekis,  
    char galutinioBudas)
```

5.9.2.5 nuskaitytiGeneruotusFailus()

```
void nuskaitytiGeneruotusFailus (  
    const std::string & failoPavadinimas,  
    Vector< student > & grupe,  
    int pKiekis,  
    char galutinioBudas)
```

5.9.2.6 penktas()

```
void penktas (  
    int pKiekis)
```

5.9.2.7 pirmas()

```
void pirmas (  
    Vector< student > & grupe,  
    char spausBudas,  
    char rusiavimoBudas,  
    char galutinioBudas,  
    int pKiekis)
```

5.9.2.8 rusiuotiOutput()

```
void rusiuotiOutput (
    Vector< student > & grupe,
    char rusiavimoBudas,
    char galutinioBudas)
```

5.9.2.9 septintas()

```
void septintas (
    char galutinioBudas)
```

5.9.2.10 sestas()

```
void sestas (
    Vector< student > & grupe,
    Vector< student > & testGrupe,
    Vector< student > & nepazangus,
    Vector< student > & normalus,
    char galutinioBudas,
    char rusiavimoBudas,
    int pKiekis)
```

5.9.2.11 skirstytiStudentus()

```
void skirstytiStudentus (
    Vector< student > & grupe,
    Vector< student > & nepazangus,
    Vector< student > & normalus,
    char galutinioBudas)
```

5.9.2.12 sortedStudentSpausdinimas()

```
void sortedStudentSpausdinimas (
    std::string lowGradeFailas,
    std::string highGradeFailas,
    Vector< student > & nepazangus,
    Vector< student > & normalus,
    char galutinioBudas)
```

5.9.2.13 spausdinimas()

```
void spausdinimas (
    char spausBudas,
    char rusiavimoBudas,
    char galutinioBudas,
    Vector< student > & grupe)
```

5.9.2.14 spausdinimasFaile()

```
void spausdinimasFaile (  
    const Vector< student > & grupe,  
    char galutinioBudas)
```

5.9.2.15 spausdinimasTerminale()

```
void spausdinimasTerminale (  
    const Vector< student > & grupe,  
    char galutinioBudas)
```

5.9.2.16 trecias()

```
void trecias (  
    Vector< student > & grupe,  
    const Vector< std::string > & vardai,  
    const Vector< std::string > & pavardes,  
    char spausBudas,  
    char rusiavimoBudas,  
    char galutinioBudas,  
    int pKiekis)
```

5.9.3 Variable Documentation

5.9.3.1 programasLaikas

```
double programasLaikas [extern]
```

5.10 vector/source/main.cpp File Reference

This file handles program's flow.

```
#include <iostream>  
#include <vector>  
#include <ctime>  
#include <limits>  
#include <cstdlib>  
#include <chrono>  
#include <sstream>  
#include <fstream>  
#include <algorithm>  
#include "../include/student.h"  
#include "../include/functions.h"  
#include <iomanip>  
#include "../include/utils.h"  
#include "../include/vector.h"
```

Functions

- int `main` ()

Variables

- double `programosLaikas` = 0.0

5.10.1 Detailed Description

This file handles program's flow.

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.10.2 Function Documentation

5.10.2.1 `main()`

```
int main ()
```

5.10.3 Variable Documentation

5.10.3.1 `programosLaikas`

```
double programosLaikas = 0.0
```

5.11 vector/source/student.cpp File Reference

this file contains class methods implementation

```
#include "student.h"
#include <algorithm>
#include <numeric>
#include <iomanip>
```

Functions

- `std::ostream & operator<<` (`std::ostream &os`, `const student &studentas`)
- `std::istream & operator>>` (`std::istream &in`, `student &studentas`)

5.11.1 Detailed Description

this file contains class methods implementation

Author

Narbas

Version

v2.0

Date

2025-05-07

Copyright

Copyright (c) 2025

5.11.2 Function Documentation

5.11.2.1 `operator<<()`

```
std::ostream & operator<< (  
    std::ostream & os,  
    const student & studentas)
```

5.11.2.2 `operator>>()`

```
std::istream & operator>> (  
    std::istream & in,  
    student & studentas)
```


Index

- ~Human
 - Human, [8](#)
- ~Vector
 - Vector< T >, [15](#)
- addPazymys
 - student, [11](#)
- antras
 - functions.cpp, [36](#)
 - functions.h, [22](#)
- assign
 - Vector< T >, [15](#)
- astuntas
 - functions.cpp, [36](#)
 - functions.h, [22](#)
- at
 - Vector< T >, [15](#), [16](#)
- back
 - Vector< T >, [16](#)
- begin
 - Vector< T >, [16](#)
- capacity
 - Vector< T >, [16](#)
- capacity_
 - Vector< T >, [20](#)
- clear
 - Vector< T >, [16](#)
- const_iterator
 - Vector< T >, [14](#)
- data
 - Vector< T >, [20](#)
- egzaminoRezultatas
 - student, [12](#)
- emplace_back
 - Vector< T >, [16](#)
- empty
 - Vector< T >, [17](#)
- end
 - Vector< T >, [17](#)
- erase
 - Vector< T >, [17](#)
- front
 - Vector< T >, [17](#)
- functions.cpp
 - antras, [36](#)
 - astuntas, [36](#)
 - generuotiFaila, [36](#)
 - ketvirtas, [36](#)
 - nuskaitytiGeneruotusFailus, [36](#)
 - penktas, [36](#)
 - pirmas, [36](#)
 - programosLaikas, [38](#)
 - rusiuotiOutput, [36](#)
 - septintas, [37](#)
 - sestas, [37](#)
 - skirstytiStudentus, [37](#)
 - sortedStudentSpausdinimas, [37](#)
 - spausdinimas, [37](#)
 - spausdinimasFaile, [37](#)
 - spausdinimasTerminale, [38](#)
 - trecias, [38](#)
- functions.h
 - antras, [22](#)
 - astuntas, [22](#)
 - generuotiFaila, [22](#)
 - ketvirtas, [22](#)
 - nuskaitytiGeneruotusFailus, [22](#)
 - penktas, [23](#)
 - pirmas, [23](#)
 - rusiuotiOutput, [23](#)
 - septintas, [23](#)
 - sestas, [23](#)
 - skirstytiStudentus, [23](#)
 - sortedStudentSpausdinimas, [24](#)
 - spausdinimas, [24](#)
 - spausdinimasFaile, [24](#)
 - spausdinimasTerminale, [24](#)
 - trecias, [24](#)
- galutinisM
 - student, [12](#)
- galutinisV
 - student, [13](#)
- generuotiFaila
 - functions.cpp, [36](#)
 - functions.h, [22](#)
- getEgzaminoRezultatas
 - student, [11](#)
- getGalutinisM
 - student, [11](#)
- getGalutinisV
 - student, [11](#)
- getPavarde
 - Human, [8](#)
- getPazymiai
 - student, [11](#)

getVardas
 Human, 8

Human, 7
 ~Human, 8
 getPavarde, 8
 getVardas, 8
 Human, 7
 pavarde, 8
 setPavarde, 8
 setVardas, 8
 skaiciuotiGalutini, 8
 vardas, 8

insert
 Vector< T >, 17

iterator
 Vector< T >, 14

ketvirtas
 functions.cpp, 36
 functions.h, 22

main
 main.cpp, 39

main.cpp
 main, 39
 programosLaikas, 39

nuskaitytiGeneruotusFailus
 functions.cpp, 36
 functions.h, 22

operator!=
 Vector< T >, 17

operator<
 Vector< T >, 20
 vector.h, 30

operator<<
 student, 12
 student.cpp, 40
 student.h, 26

operator<=
 vector.h, 30

operator>
 vector.h, 30

operator>>
 student, 12
 student.cpp, 40
 student.h, 26

operator>=
 vector.h, 30

operator=
 student, 11
 Vector< T >, 18

operator==
 Vector< T >, 18

operator[]
 Vector< T >, 18

pavarde
 Human, 8

pazymiai
 student, 13

penktas
 functions.cpp, 36
 functions.h, 23

pirmas
 functions.cpp, 36
 functions.h, 23

pop_back
 Vector< T >, 18

programosLaikas
 functions.cpp, 38
 main.cpp, 39

push_back
 Vector< T >, 18, 19

reallocate
 Vector< T >, 19

reserve
 Vector< T >, 19

resize
 Vector< T >, 19

rusiuotiOutput
 functions.cpp, 36
 functions.h, 23

septintas
 functions.cpp, 37
 functions.h, 23

sestas
 functions.cpp, 37
 functions.h, 23

setEgzaminoRezultatas
 student, 11

setGalutinisM
 student, 11

setGalutinisV
 student, 11

setPavarde
 Human, 8

setPazymiai
 student, 12

setVardas
 Human, 8

shrink_to_fit
 Vector< T >, 19

size
 Vector< T >, 19

size_
 Vector< T >, 20

skaiciuotiGalutini
 Human, 8
 student, 12

skaiciuotiMed
 student, 12

skaiciuotiVid
 student, 12

- skirstytiStudentus
 - functions.cpp, 37
 - functions.h, 23
- sortedStudentSpausdinimas
 - functions.cpp, 37
 - functions.h, 24
- spausdinimas
 - functions.cpp, 37
 - functions.h, 24
- spausdinimasFaile
 - functions.cpp, 37
 - functions.h, 24
- spausdinimasTerminale
 - functions.cpp, 38
 - functions.h, 24
- student, 9
 - addPazymys, 11
 - egzaminoRezultatas, 12
 - galutinisM, 12
 - galutinisV, 13
 - getEgzaminoRezultatas, 11
 - getGalutinisM, 11
 - getGalutinisV, 11
 - getPazymiai, 11
 - operator<<, 12
 - operator>>, 12
 - operator=, 11
 - pazymiai, 13
 - setEgzaminoRezultatas, 11
 - setGalutinisM, 11
 - setGalutinisV, 11
 - setPazymiai, 12
 - skaiciuotiGalutini, 12
 - skaiciuotiMed, 12
 - skaiciuotiVid, 12
 - student, 10
- student.cpp
 - operator<<, 40
 - operator>>, 40
- student.h
 - operator<<, 26
 - operator>>, 26
- swap
 - Vector< T >, 19
 - vector.h, 30
- tikrintiInput
 - utils.h, 29
- trecias
 - functions.cpp, 38
 - functions.h, 24
- utils.h
 - tikrintiInput, 29
- vardas
 - Human, 8
- Vector
 - Vector< T >, 15
- Vector< T >, 13
 - ~Vector, 15
 - assign, 15
 - at, 15, 16
 - back, 16
 - begin, 16
 - capacity, 16
 - capacity_, 20
 - clear, 16
 - const_iterator, 14
 - data, 20
 - emplace_back, 16
 - empty, 17
 - end, 17
 - erase, 17
 - front, 17
 - insert, 17
 - iterator, 14
 - operator!=, 17
 - operator<, 20
 - operator=, 18
 - operator==, 18
 - operator[], 18
 - pop_back, 18
 - push_back, 18, 19
 - reallocate, 19
 - reserve, 19
 - resize, 19
 - shrink_to_fit, 19
 - size, 19
 - size_, 20
 - swap, 19
 - Vector, 15
- vector.h
 - operator<, 30
 - operator<=, 30
 - operator>, 30
 - operator>=, 30
 - swap, 30
- vector/include/functions.h, 21, 25
- vector/include/student.h, 25, 26
- vector/include/utils.h, 28, 29
- vector/include/vector.h, 29, 31
- vector/source/functions.cpp, 34
- vector/source/main.cpp, 38
- vector/source/student.cpp, 39