OBJ_Uzduotys3

Generated by Doxygen 1.10.0

Chapter 1

Obj_Uzduotys3

1.1 Spartos analizė 1

| Test 1 |

Elementu skaičius	std::vector	Vector_Lib
10000	0.0001812 s	0.000102 s
100000	0.0014087 s	0.000753 s
1000000	0.0082862 s	0.0060813 s
10000000	0.0751286 s	0.0632606 s
100000000	0.714488 s	0.0545918 s

| Test 2 |

Elementu skaičius	std::vector	Vector_Lib
10000	0.0001056 s	0.0001116 s
100000	0.0009402 s	0.0006627 s
1000000	0.0074966 s	0.0054467 s
10000000	0.0734579 s	0.0617664 s
100000000	0.70871 s	0.541075 s

| Test3 |

Elementu skaičius	std::vector	Vector_Lib
10000	0.0001346 s	0.000109 s
100000	0.000992 s	0.0007352 s
1000000	0.0074966 s	0.0054467 s
10000000	0.0738471 s	0.0616216 s
100000000	0.699923 s	0.543998 s

1.2 Spartos analizė 2

1.2.0.1 Failo nuskaitymas

| Test 1|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.005047 s	0.004853 s
1.000.000	0.041024 s	0.047944 s
10.000.000	0.393229 s	0.440238 s

| Test 2|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.005278 s	0.004612 s
1.000.000	0.041093 s	0.051630 s
10.000.000	0.390727 s	0.409186 s

| Test 3|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.006231 s	0.006952 s
1.000.000	0.045362 s	0.044390 s
10.000.000	0.388422 s	0.435645 s

1.2.0.2 Studentų rūšiavimas į dvi grupes

| Test 1|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.042480 s	0.055253 s
1.000.000	0.400612 s	0.556447 s
10.000.000	4.133181 s	5.320045 s

| Test 2|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.040898 s	0.056640 s
1.000.000	0.401288 s	0.555743 s
10.000.000	4.057258 s	5.279137 s

| Test 3|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.041255 s	0.055249 s
1.000.000	0.400448 s	0.557170 s
10.000.000	4.072259 s	5.268330 s

1.2.0.3 Studentų išvedimas į failą

| Test 1|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.824902 s	0.684435 s
1.000.000	9.990556 s	7.903053 s
10.000.000	111.325708 s	89.824931 s

| Test 2|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.812058 s	0.705150 s
1.000.000	10.026291 s	8.043667 s
10.000.000	111.077281 s	90.445027 s

| Test 3|

Studentų skaičius	std::vector	Vector_Lib
100.000	0.827169 s	0.682701 s
1.000.000	10.082943 s	7.949102 s
10.000.000	110.856787 s	92.131147 s

1.3 System specs

Test were run on a system with these specs: `ryzen 5 5600x`, `32gb ddr4 3600mhz`, `rtx 3060 ti`, `250gb Kingston ssd`.

1.4 Vector_Lib functions

Examples for functions `push_back()`, `at()`, `operator[]`, `erase()`, `size()` used in Vector_Lib.

1.4.1 1. `push_back()`

```
#include "Vector_Lib.h"
#include "Studentas.h"

int main() {
    Vector_Lib<int> vec;

    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);

    vec.print();


    system("pause");
}
Output:
1 2 3
```

1.4.2 2. at()

```
#include "Vector_Lib.h"
#include "Studentas.h"

int main() {
    Vector_Lib<int> vec;

    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);

    std::cout « vec.at(1) « endl;


    system("pause");
}
Output:
2
#include "Vector_Lib.h"
#include "Studentas.h"

int main() {
    Vector_Lib<int> vec;

    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);

    std::cout « "Original vector: " « endl;
    vec.print();

    vec.at(1) = 8;

    std::cout « "Vector after at(1) = 8; : " « endl;
    vec.print();

    system("pause");
}
Output:
Original vector:
1 2 3
Vector after at(1) = 8; :
1 8 3
```

1.4.3 3. operator[]

```
#include "Vector_Lib.h"
#include "Studentas.h"

int main() {
    Vector_Lib<int> vec;

    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);

    std::cout « "Original vector: ";
    vec.print();
    std::cout « endl;

    std::cout « "Vector value at 3rd position : " « vec[2] « endl;


    system("pause");
}
Output:
Original vector: 1 2 3
Vector value at 3rd position : 3
```

1.4.4 4. erase()

```
#include "Vector_Lib.h"
#include "Studentas.h"

int main() {
    Vector_Lib<int> vec;

    for (int i = 0; i < 15; i++) {
        vec.push_back(i);
```

```
    }

    std::cout « "Original vector: ";
    vec.print();
    std::cout « endl;

    vec.erase(2);
    std::cout « "Vector after erasing the 3rd member: ";
    vec.print();
    std::cout « endl;

    vec.erase(vec.begin() + 4, vec.end() - 1);
    std::cout « "Vector after erasing all members accept the first four and the last: ";
    vec.print();
    std::cout « endl;


    system("pause");
}
Output:
Original vector: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Vector after erasing the 3rd member: 0 1 3 4 5 6 7 8 9 10 11 12 13 14

Vector after erasing all members accept the first four and the last: 0 1 3 4 14
```

1.4.5 5. `size`

```
#include "Vector_Lib.h"
#include "Studentas.h"

int main() {
    Vector_Lib<int> vec;

    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);

    std::cout « vec.size() « endl;


    system("pause");
}
Output:
3
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

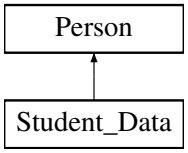
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Person Class Reference

Inheritance diagram for Person:



Public Member Functions

- **Person** (const string &name, const string &surname)
- void **SetName** (string name)
- void **SetSurname** (string surname)
- string **vardas** () const
- string **pavarde** () const
- virtual void **f** ()=0

Protected Attributes

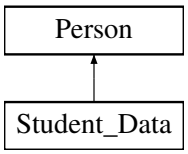
- string **student_name**
- string **student_surname**

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

- Person.h

5.2 Student_Data Class Reference

Inheritance diagram for Student_Data:



Public Member Functions

- **Student_Data** (const string &name, const string &surname, double grade, const Vector_Lib< double > &HW_)
- **Student_Data** (istream &is)
- string **vardas** () const
- string **pavarde** () const
- double **egzaminas** () const
- Vector_Lib< double > **ND** () const
- void **SetName** (string name)
- void **SetSurname** (string surname)
- void **SetExam** (double grade)
- void **SetHW** (const Vector_Lib< double > &HW_)
- istream & **readStudent** (istream &)
- **Student_Data** (const Student_Data &Adata)
- Student_Data & **operator=** (const Student_Data &Adata)
- **Student_Data** (Student_Data &&Adata) noexcept
- Student_Data & **operator=** (Student_Data &&Adata) noexcept
- void f ()

Public Member Functions inherited from Person

- **Person** (const string &name, const string &surname)
- void **SetName** (string name)
- void **SetSurname** (string surname)
- string **vardas** () const
- string **pavarde** () const

Additional Inherited Members

Protected Attributes inherited from Person

- string **student_name**
- string **student_surname**

5.2.1 Member Function Documentation

5.2.1.1 f()

```
void Student_Data::f ( )  [virtual]
```

Implements Person.

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

- Studentas.h
- Studentas.cpp

5.3 Vector_Lib< V_Lib > Class Template Reference

Public Types

- using **iterator** = V_Lib*
- using **const_iterator** = const V_Lib*
- typedef V_Lib **value_type**
- typedef value_type & **reference**
- typedef const value_type & **const_reference**
- typedef value_type * **pointer**
- typedef const value_type * **const_pointer**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef size_t **size_type**

Public Member Functions

- **Vector_Lib** (int capacity)
- **Vector_Lib** (const Vector_Lib &Adata)
- Vector_Lib< V_Lib > & **operator=** (const Vector_Lib &Adata)
- **Vector_Lib** (Vector_Lib &&Adata) noexcept
- Vector_Lib< V_Lib > & **operator=** (Vector_Lib &&Adata) noexcept
- void **push_back** (V_Lib data)
- void **pop_back** ()

  Adds a new value to the back of the array.
- void **clear** ()

  Removes the last element.
- void **erase** (size_t pos)

  Deletes all array members.
- template<typename InputIt >
  void **erase** (InputIt first, InputIt last)
- void **insert** (size_t pos, const V_Lib &stuff)
- void **insert** (size_t pos, int number, const V_Lib &stuff)
- template<class InputIt >
  void **insert** (const size_t pos, InputIt first, InputIt last)
- void **resize** (size_t size)

- void **resize** (size_t size, size_t filler)

  Changes the capacity of the array and fills it with value 0.
- template<class InputIt >
  void **append_range** (InputIt first, InputIt last)

  Changes the capacity of the array and fills it with the given value.
- void **swap** (Vector_Lib &vector)
- int **size** () const

  Swaps values of the given arrays.
- int **capacity** () const

  Returns the current size of the array.
- bool **empty** () const

  Returns the capacity of the array.
- void **reserve** (int size)

  Checks if the array is empty.
- void **shrink_to_fit** ()

  Reserves space for the array by increasing its capacity.
- V_Lib **front** () const
- V_Lib **back** () const

  Returns the current element at the front of the array.
- V_Lib & **at** (int index)

  Returns the last element in the array.
- const V_Lib & **at** (int index) const

  Access the specified element in the array.
- V_Lib & **operator[ ]** (int index)
- V_Lib * **data** () const

  Access the specified element in the array.
- iterator **begin** ()

  Copys the array.
- const_iterator **begin** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **rbegin** ()
- iterator **rend** ()
- const_iterator **cbegin** () const
- const_iterator **cend** () const
- void **print** ()

Static Public Member Functions

- static size_t **max_size** ()

  Allocates only as much space for the array as it is needed based on the current number of the elements n the array.

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

- Vector_Lib.h
- Vector_Lib.cpp

Chapter 6

File Documentation

6.1 headers_vector.h

```
00001 #ifndef HEADERS_H
00002 #define HEADERS_H
00003
00004 #include "Studentas.h"
00005 #include "includes.h"
00006
00007 int Number_Of_Students;
00008 int Number_Of_Homework;
00009 string gen_s;
00010 string input_mode;
00011
00012
00013 void splitstudents(Vector_Lib<Student_Data>& S_Data, string mode);
00014 bool isDigit(const string& str_placeholder, int check);
00015 bool isString(const string& str_placeholder);
00016 void printData(const Vector_Lib<Student_Data>& Sdata, string mode, string filename);
00017 int fileInput(const string filename);
00018 int generateFile();
00019 int manualInput();
00020 void Input(Student_Data& Sdata, string gen_s);
00021 //Deklaruotos funkcijos
00022
00023 #endif
```

6.2 includes.h

```
00001 #ifndef INCLUDES_H
00002 #define INCLUDES_H
00003
00004 #include <iostream>
00005 #include "Vector_Lib.h"
00006 #include <iomanip>
00007 #include <string>
00008 #include <algorithm>
00009 #include <random>
00010 #include <sstream>
00011 #include <chrono>
00012 #include <fstream>
00013 #include <list>
00014 #include <deque>
00015 #include <cstdio>
00016 #include <utility>
00017 #include <bits/stdc++.h>
00018
00019 using namespace std;
00020 using namespace std::chrono;
00021
00022 #endif
```

6.3 Person.h

```
00001 #ifndef PERSON_H
00002 #define PERSON_H
00003
00004 #include "includes.h"
00005
00006 class Person {
00007 protected:
00008     string student_name;
00009     string student_surname;
00010 public:
00011     Person() : student_name("") , student_surname("") {}
00012     Person(const string& name, const string& surname) : student_name(name), student_surname(surname)
      {}
00013     //Konstruktoriai
00014
00015     void SetName(string name){ student_name = name; }
00016     void SetSurname(string surname){ student_surname = surname; }
00017     //Setters
00018
00019     string vardas() const { return student_name;}
00020     string pavarde() const { return student_surname;}
00021     //Getters
00022
00023     virtual void f() = 0;
00024     //Padaro bazine klase Person abstrakcija
00025
00026     ~Person() {}
00027     //Deconstructor
00028 };
00029
00030 #endif
```

6.4 Studentas.h

```
00001 #ifndef STUDENTAS_H
00002 #define STUDENTAS_H
00003
00004 #include "includes.h"
00005 #include "utility"
00006 #include "Person.h"
00007
00008 class Student_Data : public Person {
00009 private:
00010     double exam;
00011     Vector_Lib<double> HW;
00012 public:
00013     Student_Data() : Person(), exam(0) { }
00014     Student_Data(const string& name, const string& surname, double grade, const Vector_Lib<double>&
      HW_) : Person(student_name, student_surname), exam(grade), HW(HW_) {}
00015     Student_Data(istream& is);
00016     //Constructors
00017
00018     string vardas() const { return student_name;}
00019     string pavarde() const { return student_surname;}
00020     double egzaminas() const { return exam;}
00021     Vector_Lib<double> ND() const { return HW;}
00022     //Getters
00023
00024     void SetName(string name){ student_name = name; }
00025     void SetSurname(string surname){ student_surname = surname; }
00026     void SetExam( double grade){ exam = grade; }
00027     void SetHW (const Vector_Lib<double>& HW_) { HW = HW_; }
00028     //Setters
00029
00030     istream& readStudent(istream&);
00031
00032     ~Student_Data(); //Deconstructor
00033     Student_Data(const Student_Data& Adata); //Copy constructor
00034     Student_Data& operator=(const Student_Data& Adata);   //Copy assigment operator
00035     Student_Data(Student_Data&& Adata) noexcept ;    //Move constructor
00036     Student_Data& operator=(Student_Data&& Adata) noexcept;    //Move assigment operator
00037     //Rule of five
00038
00039     void f();
00040     //Pavercia derived klase Student_Data neabstrakcia, nes ji is bazines klases Person paveldi
      abstraktuma
00041 };
00042
00043 double avg_grade(const Student_Data& Sdata);
00044 double median_grade(const Student_Data& Sdata);
00045 //function declarations
```

```
00046
00047 istream& operator»(istream& set, Student_Data& Sdata);
00048 ostream& operator«(ostream& print, Student_Data Sdata);
00049 //Input/Output operators
00050 #endif
```

6.5 Vector_Lib.h

```
00001 #ifndef VECTOR_LIB_H
00002 #define VECTOR_LIB_H
00003
00004 #include <iostream>
00005 #include <limits>
00006 #include <algorithm>
00007 #include <memory>
00008 #include <iterator>
00009
00010 using namespace std;
00011
00012 template <typename V_Lib>
00013
00014 class Vector_Lib {
00015     V_Lib* arr;
00016     int Capacity;
00017     int current;
00018
00019 public:
00020     using iterator = V_Lib*;
00021     using const_iterator = const V_Lib*;
00022
00023     // Member Types ----------------------------------------------------------
00024     typedef V_Lib value_type;
00025     typedef value_type& reference;
00026     typedef const value_type& const_reference;
00027     typedef value_type* pointer;
00028     typedef const value_type* const_pointer;
00029     typedef std::reverse_iterator<iterator> reverse_iterator;
00030     typedef std::reverse_iterator<const_iterator> const_reverse_iterator;
00031     typedef ptrdiff_t difference_type;
00032     typedef size_t size_type;
00033
00034     Vector_Lib(){arr = new V_Lib[1]; Capacity = 1; current = 0;} // Constructor
00035     explicit Vector_Lib(int capacity) : arr(new V_Lib[capacity]), Capacity(capacity), current(0) {}
00036     ~Vector_Lib() { delete[] arr; } // Deconstructor
00037     Vector_Lib(const Vector_Lib& Adata) : Capacity(Adata.Capacity), current(Adata.current) {
00038         arr = new V_Lib[Capacity];
00039
00040         for (int i = 0; i < current; i++) {
00041             arr[i] = Adata.arr[i];
00042         }
00043     }; // Copy constructor
00044     Vector_Lib<V_Lib>& operator=(const Vector_Lib& Adata); // Copy assignment
00045     Vector_Lib(Vector_Lib&& Adata) noexcept; // Move constructor
00046     Vector_Lib<V_Lib>& operator=(Vector_Lib&& Adata) noexcept; //Move assignment
00047
00048     // Modifiers functions ------------------------------------------------- 8/11
00049     void push_back(V_Lib data) {
00050         if (current == Capacity) {
00051             V_Lib* temp = new V_Lib[2 * Capacity];
00052
00053             for (int i = 0; i < Capacity; i++) {
00054                 temp[i] = arr[i];
00055             }
00056
00057             delete[] arr;
00058             Capacity *= 2;
00059             arr = temp;
00060         }
00061
00062         arr[current] = data;
00063         current++;
00064
00065     };
00066     void pop_back() {current--;};
00067     void clear() {
00068         delete[] arr;
00069         arr = new V_Lib[1];
00070         current = 0;
00071     };
00072     void erase(size_t pos); // 1/2 /// Erases a member from the array at the specified location
00073
00074     template <typename InputIt>
00075     void erase(InputIt first, InputIt last) {
```

```
00076         iterator first_value = first;
00077         iterator last_value = last;
00078 
00079         if (first_value >= arr && last_value <= arr + current) {
00080             copy(last_value, arr + current, first_value);
00081             current -= (last_value - first_value);
00082         }
00083     } // 2/2 ///Erases members from the array specified by the given iterators
00084 
00085     void insert(size_t pos, const V_Lib& stuff); // 1/3 /// Inserts an element into the array at the specified location
00086     void insert(size_t pos, int number, const V_Lib& stuff);  // 2/3 // Inserts a number of the same element at the specified location
00087 
00088     template <class InputIt>
00089     void insert(const size_t pos, InputIt first, InputIt last) {
00090         if (pos < 0 || pos > current) {
00091             throw out_of_range("Index out of range!");
00092         } else {
00093             size_t size = distance(first, last);
00094             size_t new_cap = current + size;
00095 
00096             V_Lib* temp = new V_Lib[new_cap];
00097 
00098             for (size_t i = 0; i < pos; i++) {
00099                 temp[i] = arr[i];
00100             }
00101 
00102             size_t index = pos;
00103             for (auto i = first; i != last; i++) {
00104                 temp[index++] = *i;
00105             }
00106 
00107             for (size_t i = pos; i < current; i ++) {
00108                 temp[i + size] = arr[i];
00109             }
00110 
00111             delete[] arr;
00112             arr = temp;
00113             Capacity = new_cap;
00114             current = new_cap;
00115         }
00116     }; // Ikisau cia, nes man kroniskai mete "undefined reference" nors ir buvo defined // 3/3 /// Inserts a number of element specified by the given iterators at the given position
00117 
00118     void resize(size_t size);
00119     void resize(size_t size, size_t filler);
00120 
00121     template <class InputIt>
00122     void append_range(InputIt first, InputIt last) {
00123 
00124         size_t size = distance(first, last);
00125         size_t new_cap = current + size;
00126 
00127         V_Lib* temp = new V_Lib[new_cap];
00128 
00129         for (size_t i = 0; i < current; i++) {
00130             temp[i] = arr[i];
00131         }
00132 
00133         size_t index = current;
00134         for (auto i = first; i != last; i++) {
00135             temp[index++] = *i;
00136         }
00137 
00138         delete[] arr;
00139         arr = temp;
00140         Capacity = new_cap;
00141         current = new_cap;
00142 
00143     } // reference ptsd /// Ads elements specified by the given iterators to the back of the array
00144 
00145     void swap(Vector_Lib& vector);
00146 
00147     // Capacity functions ------------------------------------------------------ 6/6
00148     int size() const {return current;};
00149     int capacity() const;
00150     bool empty() const;
00151     void reserve(int size);
00152     void shrink_to_fit();
00153     static size_t max_size();
00154     // Element access ---------------------------------------------------------- 5/5
00155     V_Lib front() const;
00156     V_Lib back() const {
00157         if (Capacity == 0) {
00158             throw out_of_range("Index out of range!");
00159         }
```

```
00160
00161         return arr[current - 1];
00162     };
00163     V_Lib& at(int index);
00164     const V_Lib& at(int index) const;
00165     V_Lib& operator[](int index) {
00166         if (index < 0 || index >= current) {
00167             throw out_of_range("Index out of range!");
00168         } else {
00169             return arr[index];
00170         }
00171     };
00172     V_Lib* data() const;
00173     // Iterators ---------------------------------------------------------- 4/4
00174
00175     iterator begin() {return arr;}
00176     const_iterator begin() const {return arr;}
00177     iterator end() {return arr + current;}
00178     const_iterator end() const {return arr + current;}
00179     iterator rbegin() {return arr + current - 1;}
00180     iterator rend() {return arr - 1;}
00181     const_iterator cbegin() const {return arr;}
00182     const_iterator cend() const {return arr + current;}
00183
00184     void print()
00185     {
00186         for (int i = 0; i < current; i++) {
00187             cout « arr[i] « " ";
00188         }
00189         cout « endl;
00190     }// delete later
00191
00192 };
00193
00194 #endif
```

Index