Kolokwium/

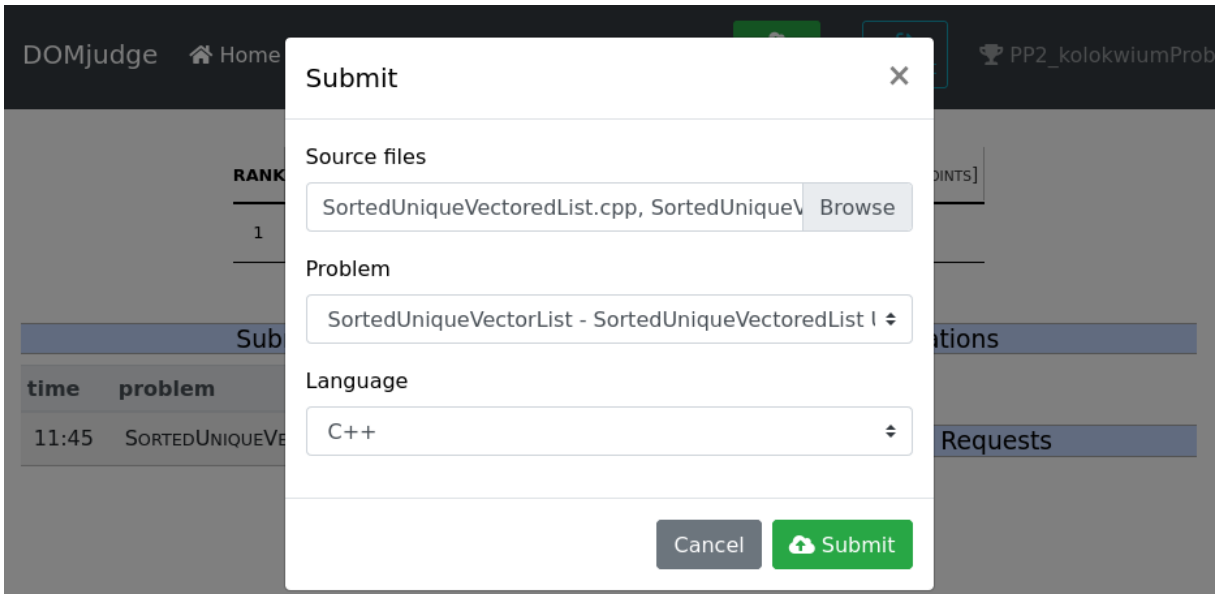
Generated by Doxygen 1.9.1

Chapter 1

Kolokwium probne z Podstaw Programowania 2 dnia 31 V 2021

1.0.1 Zasady ogolne:

1. Zanim cokolwiek Panstwo zrobia prosze o uzupelnienie swoich danych w pliku: main.cpp, dane te to:
   - FIRSTNAME
   - SURNAME
   - MAIL
   - BOOK_ID (nr albumu)
2. Kolokwium nie da się nie zdać - jest traktowane jako punkty, które się sumują do reszty, nie będzie poprawki grupowej!
3. W trakcie kolokwium nalezy miec wlaczona kamerke i nie miec sluchawek, kto będzie bez kamery to o 1/3 punktów mniej.
4. Wyslana paczka ma sie bezwzglednie kompilowac na systemie Linux.
   - Jak ktos nie ma linuxa moze uzyc narzedzia: `http://administrare.kis.agh.edu.pl:12345` (konieczny VPN AGH) i na nim zarejestrowac sie, ale jako nick powinien byc **numer albumu**, aczkolwiek prosze tez uzupelnic pozostale dane (imie, nazwisko, mail).
   - Aby wyslac zadanie nalezy wybrac konkurs (*PP2_kolokwiumProbne*), problem (*SortedUnique...*), oraz jezyk programowania (*c++*), prosze zalaczyc obydwa pliki SortedUniqueVectoredList.h, SortedUniqueVectoredList.cpp, jak na obrazku:



5. Kolokwium z zalozenia bedzie sprawdzane automatycznie, przez testy jednostkowe dostepne w pliku : SortedUniqueVectoredListTests.cpp, dlatego poza kompilowaniem prosze aby nie crashowalo na zadnym tescie, jesli tak sie dzieje to brane pod uwage jest tylko tyle testow ile ich przechodzi do momentu crasha.
6. Mam program antyplagiatowy, dlatego prosze pracowac samodzielnie! Osoby ktore udostepniaja swoje rozwiazania rowniez beda mialy kare!
7. ∗Dobrze jakby nie bylo warningow kompilacji (flagi: `-Wall -Wextra -pedantic -Werror`, a dla hardcorów jeszcze: `-Weffc++`)
8. Punkty beda odejmowane za wycieki pamieci.
9. Zakres materiału: wykłady [1, 10]
   Szczegolowa tresc kolokwium znajda Panstwo w opisie metod klasy: SortedUniqueVectoredList

Chapter 2

Todo List

Member FIRSTNAME
Uzupelnij swoje dane:

Member SortedUniqueVectoredList::allocate_new_bucket ()
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::contains (const std::string &value) const
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::copy (const SortedUniqueVectoredList &other)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::erase (const std::string &value)
zaimplementuj, szczegoly w pliku naglowkowym (opcjonalne zadanie)

Member SortedUniqueVectoredList::free ()
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::insert (const std::string &value)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::move (SortedUniqueVectoredList &&another)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator std::string () const
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator*= (const size_t howManyTimesMultiply)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator- (const SortedUniqueVectoredList &another) const
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator= (const SortedUniqueVectoredList &another)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator= (SortedUniqueVectoredList &&another)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator[] (size_t index)
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::operator[] (size_t index) const
zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::SortedUniqueVectoredList (SortedUniqueVectoredList &&source)

zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::SortedUniqueVectoredList (const SortedUniqueVectoredList &source)

zaimplementuj, szczegoly w pliku naglowkowym

Member SortedUniqueVectoredList::~SortedUniqueVectoredList ()

zaimplementuj, szczegoly w pliku naglowkowym

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

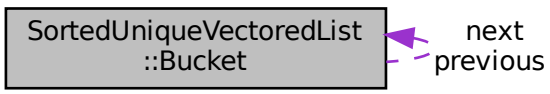
Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 SortedUniqueVectoredList::Bucket Struct Reference

Collaboration diagram for SortedUniqueVectoredList::Bucket:



Public Attributes

- std::array< std::string, BUCKET_SIZE > values
- size_t size {}
- Bucket * next = nullptr
- Bucket * previous = nullptr

Static Public Attributes

- constexpr static size_t BUCKET_SIZE = 10

5.1.1 Detailed Description

class SortedUniqueVectoredList::Bucket

Parameters

size	ilosc elementow w kubelku, tworzac pusty ma byc 0
values	elementy kubelka, jako tablica statyczna
BUCKET_SIZE	ilosc elementow w statycznej tablicy
*bucketCount_*	ilosc kubelkow
next	wskaznik na nastepny Bucket, a jesli takiego nie ma na nullptr
previous	wskaznik na poprzedni Bucket, a jesli takiego nie ma na nullptr

Note

jest to klasa zrobiona przy pomocy idiomu PIMPL, ktory polega na tym, ze w klasie zewnetrznej jest jedynie deklaracja klasy wewnetrznej, ktora jest zaimplementowana w pliku zrodlowym

Definition at line 17 of file SortedUniqueVectoredList.cpp.

5.1.2 Member Data Documentation

5.1.2.1 BUCKET_SIZE

```
constexpr static size_t SortedUniqueVectoredList::Bucket::BUCKET_SIZE = 10  [static], [constexpr]
```

Definition at line 19 of file SortedUniqueVectoredList.cpp.

5.1.2.2 next

```
Bucket* SortedUniqueVectoredList::Bucket::next = nullptr
```

Definition at line 24 of file SortedUniqueVectoredList.cpp.

5.1.2.3 previous

```
Bucket* SortedUniqueVectoredList::Bucket::previous = nullptr
```

Definition at line 25 of file SortedUniqueVectoredList.cpp.

5.1.2.4 size

```
size_t SortedUniqueVectoredList::Bucket::size {}
```

Definition at line 22 of file SortedUniqueVectoredList.cpp.

5.1.2.5 values

```
std::array<std::string, BUCKET_SIZE> SortedUniqueVectoredList::Bucket::values
```

Definition at line 21 of file SortedUniqueVectoredList.cpp.

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

- SortedUniqueVectoredList.cpp

5.2 SortedUniqueVectoredList Class Reference

Klasa SortedUniqueVectoredList, stanowiaca liste z glowa tablic dynamicznych trzymajacych teksty, ktore maja byc posortowane i unikalne. Bez obaw, nie musi byc optymalnie, schemat znajduje sie tutaj:

```
#include <SortedUniqueVectoredList.h>
```

Classes

- struct Bucket

Public Member Functions

- SortedUniqueVectoredList ()=default

  konstruktor domyslny, jego zadaniem jest ustawienie pol klasy na brak elementow
- SortedUniqueVectoredList (const SortedUniqueVectoredList &source)

  *konstruktor kopiujacy, dokonujacy **gleboka kopie**, czyli nie tylko wskazniki maja byc skopiowane, ale cale kubelki*
- SortedUniqueVectoredList (SortedUniqueVectoredList &&source)

  konstruktor przenoszacy, ktory cala zawartosc z innego kontenera przeniesie zostawiajac
- ~SortedUniqueVectoredList ()

  destruktor, ktory musi koniecznie zwolnic pamiec i inne zasoby
- SortedUniqueVectoredList & operator= (const SortedUniqueVectoredList &another)

  operator przypisania, ktory ma za zadanie skopiowac doglebnie tresc, analogicznie jak konstruktor kopiujacy SortedUniqueVectoredList(const SortedUniqueVectoredList&)
- SortedUniqueVectoredList & operator= (SortedUniqueVectoredList &&another)

  operator przypisania, ktory ma za zadanie przeniesc zawartosc z obiektu zrodlowego
- auto size () const

  Metoda zwracajaca aktualnie posiadana ilosc elementow w kontenerze.
- auto capacity () const

  Metoda zwracajaca informacje ile elementow zmiesci sie w kontenerze.
- auto bucket_count () const

  Metoda zwracajaca informacje ile kubelkow jest obecnie zaallokowanych.
- void insert (const std::string &value)

  metoda, ktora skopiuje podany tekst i umiesci na kontenerze, o ile takowego jeszcze nie ma. W razie braku miejsca kontener powinien zostac zwiekszony.
- std::string & operator[ ] (size_t index)

  operator indeksowania, ktory otrzymawszy indeks zwroci referencje do tekstu znajdujacego sie na danej pozycji w kontenerze
- const std::string & operator[ ] (size_t index) const

operator indeksowania, podobny do powyzszego operator[], ale zwraca `const string` i jest metoda stala

- SortedUniqueVectoredList operator- (const SortedUniqueVectoredList &another) const

  operator, ktory tworzy kontener zawierajacy wszystkie elementy z pierwszego kontenera, ktorych nie ma w kontenerze `another`
- SortedUniqueVectoredList & operator∗= (const size_t howManyTimesMultiply)

  operator, ktory kazdy ze skladowanych tekstow `this` zwielokrotni wskazana ilosc razy
- operator std::string () const

  operator konwersji, ktory wszystkie teksty polaczy w jeden bez jakichkolwiek separatorow
- void erase (const std::string &value)

  Opcjonalne (nie ma na to testow): metoda usuwajacy element o danej wartosci.

Protected Member Functions

- void allocate_new_bucket ()

  metoda pomocnicza allokujaca nowy Bucket na koncu listy (nowy SortedUniqueVectoredList::tail)
- void free ()

  metoda pomocnicza zwalniajaca zasoby
- void move (SortedUniqueVectoredList &&another)

  metoda pomocnicza przenoszaca zawartosc z obiektu zrodlowego na ∗this.
- void copy (const SortedUniqueVectoredList &other)

  metoda pomocnicza kopiujaca z obiektu zrodlowego na ∗this.
- bool contains (const std::string &value) const

  metoda pomocnicza zwracajaca informacje czy dany element jest juz w kontenerze.

Friends

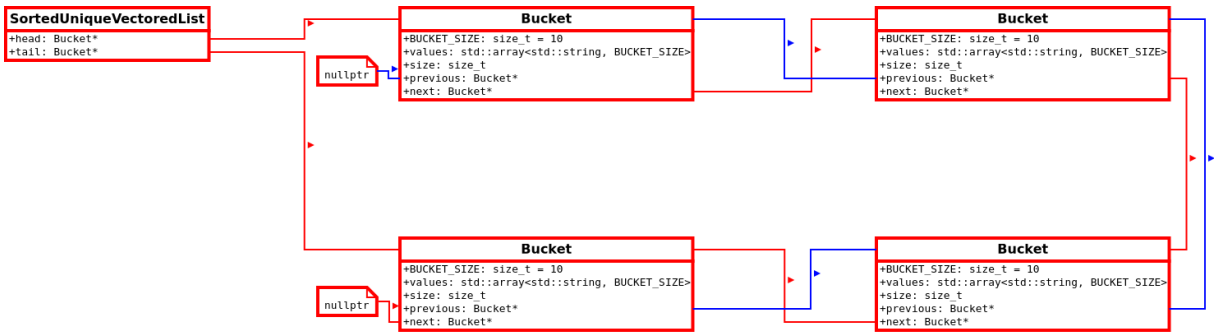
- std::ostream & operator<< (std::ostream &stream, const SortedUniqueVectoredList &container)

  Opcjonalne (nie ma na to testow)

5.2.1 Detailed Description

Klasa SortedUniqueVectoredList, stanowiaca liste z glowa tablic dynamicznych trzymajacych teksty, ktore maja byc posortowane i unikalne. Bez obaw, nie musi byc optymalnie, schemat znajduje sie tutaj:

class SortedUniqueVectoredList



Jak widac w praktyce jest to lista z glowa wezlow Bucket. Dodawane elementy sa jesli juz takowego nie ma, oraz sortowane. Kontener ten w razie braku miejsca tworzy nowy @Bucket i tam dodaje nowy element.

Note

Zmiany pliku naglowkowego nie powinny byc konieczne, prosze probowac zmienic jedynie w pliku zrodlowym SortedUniqueVectoredList.cpp.

Parameters

*size_*	ilosc elementow w kontenerze, tworzac pusty ma byc 0, po kazdym zawolaniu push_back powinno byc o jeden wiecej
*capacity_*	ilosc zaallokowanej pamieci w kontenerze na elementy, wywolania push_back w razie potrzeby maja dokonac reallokacji powiekszajacej o jeden Bucket, wtedy capacity jest zwiekszane o Bucket::BUCKET_SIZE
*bucket↩ Count_*	ilosc kubelkow
head	wskaznik na pierwszy Bucket
tail	wskaznik na pierwszy Bucket

Definition at line 32 of file SortedUniqueVectoredList.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 SortedUniqueVectoredList() [1/3]

```
SortedUniqueVectoredList::SortedUniqueVectoredList ( )  [default]
```

konstruktor domyslny, jego zadaniem jest ustawienie pol klasy na brak elementow

5.2.2.2 SortedUniqueVectoredList() [2/3]

```
SortedUniqueVectoredList::SortedUniqueVectoredList (
            const SortedUniqueVectoredList & source )
```

konstruktor kopiujacy, dokonujacy **gleboka kopie**, czyli nie tylko wskazniki maja byc skopiowane, ale cale kubelki

Parameters

source	- kontener zrodkowy, z ktorego musza byc skopiowane wszystkie dane

Note

jak dobrze zaimplementujemy metode SortedUniqueVectoredList::copy to wystarczy ja tutaj zawolac

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 29 of file SortedUniqueVectoredList.cpp.

5.2.2.3 SortedUniqueVectoredList() [3/3]

```
SortedUniqueVectoredList::SortedUniqueVectoredList (
            SortedUniqueVectoredList && source )
```

konstruktor przenoszacy, ktory cala zawartosc z innego kontenera przeniesie zostawiajac

Parameters

source	- kontener zrodkowy, z ktorego musza byc przeniesione wszystkie dane

Note

kontener zrodlowy w stanie jak po zawolaniu konstruktora bezargumentowego

jak dobrze zaimplementujemy metody SortedUniqueVectoredList::move i SortedUniqueVectoredList::free to warto je tutaj zawolac

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 35 of file SortedUniqueVectoredList.cpp.

5.2.2.4 ~SortedUniqueVectoredList()

```
SortedUniqueVectoredList::~SortedUniqueVectoredList ( )
```

destruktor, ktory musi koniecznie zwolnic pamiec i inne zasoby

Note

jak dobrze zaimplementujemy metode SortedUniqueVectoredList::free to wystarczy ja tutaj zawolac

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 41 of file SortedUniqueVectoredList.cpp.

5.2.3 Member Function Documentation

5.2.3.1 allocate_new_bucket()

```
void SortedUniqueVectoredList::allocate_new_bucket ( )  [protected]
```

metoda pomocnicza allokujaca nowy Bucket na koncu listy (nowy SortedUniqueVectoredList::tail)

Note

prosze pamietac o SortedUniqueVectoredList::bucketCount_ SortedUniqueVectoredList::capacity_

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 75 of file SortedUniqueVectoredList.cpp.

5.2.3.2 bucket_count()

```
auto SortedUniqueVectoredList::bucket_count ( ) const  [inline]
```

Metoda zwracajaca informacje ile kubelkow jest obecnie zaallokowanych.

Returns

wartosc `bucketCount_`

Definition at line 97 of file SortedUniqueVectoredList.h.

5.2.3.3 capacity()

```
auto SortedUniqueVectoredList::capacity ( ) const  [inline]
```

Metoda zwracajaca informacje ile elementow zmiesci sie w kontenerze.

Returns

wartosc `capacity_`

Definition at line 90 of file SortedUniqueVectoredList.h.

5.2.3.4 contains()

```
bool SortedUniqueVectoredList::contains (
            const std::string & value ) const  [protected]
```

metoda pomocnicza zwracajaca informacje czy dany element jest juz w kontenerze.

nie musi byc optymalna, moze isc sekwencyjnie po wszystkich elementach

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 100 of file SortedUniqueVectoredList.cpp.

5.2.3.5 copy()

```
void SortedUniqueVectoredList::copy (
            const SortedUniqueVectoredList & other )  [protected]
```

metoda pomocnicza kopiujaca z obiektu zrodlowego na ∗this.

Note

nalezy pamietac o zwalnianiu zasobow

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 94 of file SortedUniqueVectoredList.cpp.

5.2.3.6 erase()

```
void SortedUniqueVectoredList::erase (
            const std::string & value )
```

Opcjonalne (nie ma na to testow): metoda usuwajacy element o danej wartosci.

Todo zaimplementuj, szczegoly w pliku naglowkowym (opcjonalne zadanie)

Definition at line 63 of file SortedUniqueVectoredList.cpp.

5.2.3.7 free()

```
void SortedUniqueVectoredList::free ( )  [protected]
```

metoda pomocnicza zwalniajaca zasoby

Note

nie tylko SortedUniqueVectoredList::head i SortedUniqueVectoredList::tail powinny byc zwolnione, ale rowniez wszystko pomiedzy

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 83 of file SortedUniqueVectoredList.cpp.

5.2.3.8 insert()

```
void SortedUniqueVectoredList::insert (
            const std::string & value )
```

metoda, ktora skopiuje podany tekst i umiesci na kontenerze, o ile takowego jeszcze nie ma. W razie braku miejsca kontener powinien zostac zwiekszony.

Parameters

text2Add	- tekst do skopiowania doglebnie (na nowa dynamiczna pamiec)

Postcondition

Dodany tekst zostanie skopiowany i umieszczony na koncu kontenera. W razie potrzeby powinien byc zaalokowany kolejny Bucket.

Note

Nie musi byc optymalnie - mozna pojechac po calym kontenerze kazdorazowo i caly kontener sortowac po kazdym wstawieniu.

Sugeruje przed wstawieniem elementu sprawdzic czy takowy jest juz w kontenerze.

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 57 of file SortedUniqueVectoredList.cpp.

5.2.3.9 move()

```
void SortedUniqueVectoredList::move (
            SortedUniqueVectoredList && another )  [protected]
```

metoda pomocnicza przenoszaca zawartosc z obiektu zrodlowego na ∗this.

Note

nalezy pamietac o zwalnianiu zasobow

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 88 of file SortedUniqueVectoredList.cpp.

5.2.3.10 operator std::string()

```
SortedUniqueVectoredList::operator std::string ( ) const  [explicit]
```

operator konwersji, ktory wszystkie teksty polaczy w jeden bez jakichkolwiek separatorow

Returns

tekst zawierajacy wszystkie teksty

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 69 of file SortedUniqueVectoredList.cpp.

5.2.3.11 operator*=()

```
SortedUniqueVectoredList & SortedUniqueVectoredList::operator*= (
            const size_t howManyTimesMultiply )
```

operator, ktory kazdy ze skladowanych tekstow `this` zwielokrotni wskazana ilosc razy

Note

jest to operator modyfikujacy obiekt na rzecz ktorego jest wywolywany

Parameters

howManyTimesMultiply	ile razy kazdy tekst ma byc zwielokrotniony

Returns

`*this` ze zwielokrotnionymi tekstami

Przykladowo:

```
SortedUniqueVectoredList a;
a.insert("HejHa");
a *= 3;
// a zawiera "HejHaHejHaHejHa"
a *= 0;
// a zawiera ""
```

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 115 of file SortedUniqueVectoredList.cpp.

5.2.3.12 operator-()

```
SortedUniqueVectoredList SortedUniqueVectoredList::operator- (
            const SortedUniqueVectoredList & another ) const
```

operator, ktory tworzy kontener zawierajacy wszystkie elementy z pierwszego kontenera, ktorych nie ma w kontenerze `another`

Parameters

another	kontener, ktorego elementy maja byc usuniete z `this` o ile w nim sa

Returns

Nowo-utworzony kontener zawierajacy wszystkie elementy z `*this`, ktorych nie ma w `another`

Sugeruje utworzyc nowy kontener i wolac na nim insert, dla elementow bedacych w this ale nie bedacych w `another`

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 107 of file SortedUniqueVectoredList.cpp.

5.2.3.13 operator=() [1/2]

```
SortedUniqueVectoredList & SortedUniqueVectoredList::operator= (
            const SortedUniqueVectoredList & another )
```

operator przypisania, ktory ma za zadanie skopiowac doglebnie tresc, analogicznie jak konstruktor kopiujacy SortedUniqueVectoredList(const SortedUniqueVectoredList&)

Note

prosze sie upewnic, ze zadziala przypisanie na samego siebie:
```
SortedUniqueVectoredList a;
SortedUniqueVectoredList& b = a;
a = b;
```

prosze sie upewnic, ze zadziala przypisanie kaskadowe:
```
SortedUniqueVectoredList a, b, c;
a = b = c;
```

Operator przypisania powinien **zwolnic pamiec** w razie potrzeby, aby nie dopuscic do wyciekow pamieci. Mozna uzyc idiomu copy&swap.

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 134 of file SortedUniqueVectoredList.cpp.

5.2.3.14 operator=() [2/2]

```
SortedUniqueVectoredList & SortedUniqueVectoredList::operator= (
            SortedUniqueVectoredList && another )
```

operator przypisania, ktory ma za zadanie przeniesc zawartosc z obiektu zrodlowego

Note

prosze sie upewnic, ze zadziala przypisanie na samego siebie:
```
SortedUniqueVectoredList a;
SortedUniqueVectoredList& b = a;
a = std::move(b);
```

Operator przypisania przenoszacy powinien **zwolnic dotychczasowa pamiec**, aby nie dopuscic do wyciekow pamieci. Powinien tez zostawic obiekt zrodlowy w stanie jak po zawolaniu konstruktora domyslnego. Jak dobrze zaimplementujemy SortedUniqueVectoredList::move i SortedUniqueVectoredList::free to warto je zawolac

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 47 of file SortedUniqueVectoredList.cpp.

5.2.3.15 operator[]() [1/2]

```
string & SortedUniqueVectoredList::operator[] (
            size_t index )
```

operator indeksowania, ktory otrzymawszy indeks zwroci referencje do tekstu znajdujacego sie na danej pozycji w kontenerze

Parameters

index	elementu tekstowego w kontenerze

Exceptions

w	razie podania
std::out_of_range	w razie, gdy `index >= size_`

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 122 of file SortedUniqueVectoredList.cpp.

5.2.3.16 operator[]() [2/2]

```
const string & SortedUniqueVectoredList::operator[] (
            size_t index ) const
```

operator indeksowania, podobny do powyzszego operator[], ale zwraca `const string` i jest metoda stala

Todo zaimplementuj, szczegoly w pliku naglowkowym

Definition at line 128 of file SortedUniqueVectoredList.cpp.

5.2.3.17 size()

```
auto SortedUniqueVectoredList::size ( ) const  [inline]
```

Metoda zwracajaca aktualnie posiadana ilosc elementow w kontenerze.

Returns

wartosc `size_`

Definition at line 83 of file SortedUniqueVectoredList.h.

5.2.4 Friends And Related Function Documentation

5.2.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & stream,
            const SortedUniqueVectoredList & container )  [friend]
```

Opcjonalne (nie ma na to testow)

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

- SortedUniqueVectoredList.h
- SortedUniqueVectoredList.cpp

Chapter 6

File Documentation

6.1 CMakeLists.txt File Reference

Functions

- cmake_minimum_required (VERSION 3.19) project(kolokwium_probne_dlaStudentow) set(CMAKE_CXX_↩ STANDARD 14) add_subdirectory(tests) add_executable($
- main cpp SortedUniqueVectoredList cpp add_custom_target (run COMMAND ${PROJECT_NAME} DEPENDS ${PROJECT_NAME} WORKING_DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}) add_↩ custom_target(valgrind_$

6.1.1 Function Documentation

6.1.1.1 add_custom_target()

```
main cpp SortedUniqueVectoredList cpp add_custom_target (
             run COMMAND ${PROJECT_NAME} DEPENDS ${PROJECT_NAME} WORKING_DIRECTORY ${CMAKE_↩
CURRENT_BINARY_DIR}  )
```

Definition at line 8 of file CMakeLists.txt.

6.1.1.2 cmake_minimum_required()

```
cmake_minimum_required (
             VERSION 3.  19 )
```

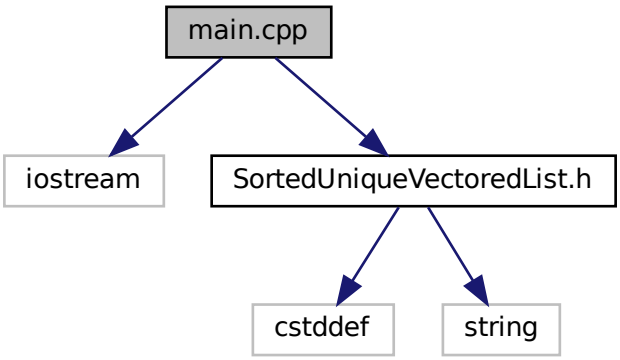
Definition at line 1 of file CMakeLists.txt.

6.2 input.txt File Reference

6.3 main.cpp File Reference

```
#include <iostream>
#include "SortedUniqueVectoredList.h"
```

Include dependency graph for main.cpp:



Functions

- void validateStudentsInfo ()
- int main ()
- constexpr size_t compileTimeStrlen (const char *text) noexcept
- constexpr size_t compileTimeCountFirstDigits (const char *text) noexcept
- constexpr bool compileTimeIsDigit (const char *text) noexcept

Variables

- constexpr const char *const FIRSTNAME = ""
- constexpr const char *const SURNAME = ""
- constexpr const char *const MAIL = ""
- constexpr const char *const BOOK_ID = ""
- constexpr const char *const teacherMail = "bazior[at]agh.edu.pl"

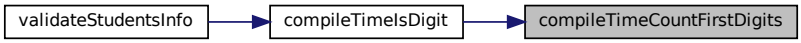
6.3.1 Function Documentation

6.3.1.1 compileTimeCountFirstDigits()

```
constexpr size_t compileTimeCountFirstDigits (
            const char * text )  [inline], [constexpr], [noexcept]
```

Definition at line 57 of file main.cpp.

Here is the caller graph for this function:

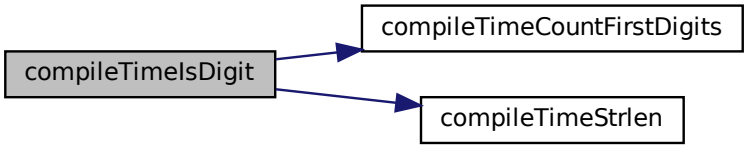


6.3.1.2 compileTimeIsDigit()

```
constexpr bool compileTimeIsDigit (
            const char * text )  [inline], [constexpr], [noexcept]
```

Definition at line 62 of file main.cpp.

Here is the call graph for this function:



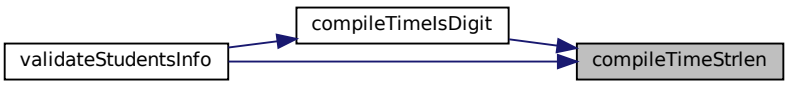
Here is the caller graph for this function:

6.3.1.3 compileTimeStrlen()

```
constexpr size_t compileTimeStrlen (
            const char * text )  [inline], [constexpr], [noexcept]
```

Definition at line 52 of file main.cpp.

Here is the caller graph for this function:



6.3.1.4 main()

```
int main ( )
```

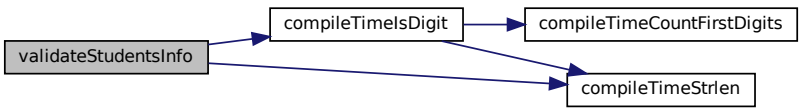
Definition at line 45 of file main.cpp.

6.3.1.5 validateStudentsInfo()

```
void validateStudentsInfo ( )
```

Definition at line 67 of file main.cpp.

Here is the call graph for this function:



6.3.2 Variable Documentation

6.3.2.1 BOOK_ID

```
constexpr const char* const BOOK_ID = ""  [constexpr]
```

Definition at line 39 of file main.cpp.

6.3.2.2 FIRSTNAME

```
constexpr const char* const FIRSTNAME = ""  [constexpr]
```

Todo Uzupelnij swoje dane:

Definition at line 36 of file main.cpp.

6.3.2.3 MAIL

```
constexpr const char* const MAIL = ""  [constexpr]
```

Definition at line 38 of file main.cpp.

6.3.2.4 SURNAME

```
constexpr const char* const SURNAME = ""  [constexpr]
```

Definition at line 37 of file main.cpp.

6.3.2.5 teacherMail

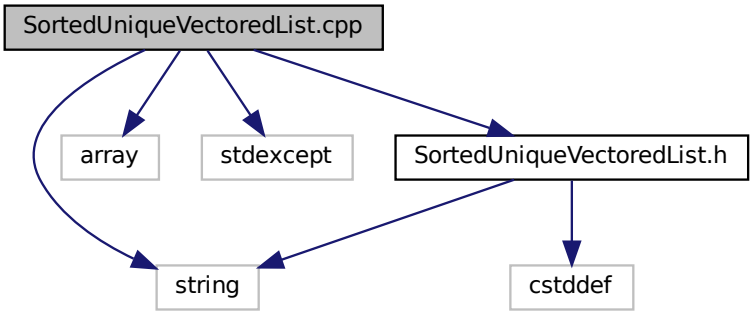
```
constexpr const char* const teacherMail = "bazior[at]agh.edu.pl"  [constexpr]
```

Definition at line 40 of file main.cpp.

6.4 SortedUniqueVectoredList.cpp File Reference

```
#include <string>
#include <array>
#include <stdexcept>
#include "SortedUniqueVectoredList.h"
```

Include dependency graph for SortedUniqueVectoredList.cpp:



Classes

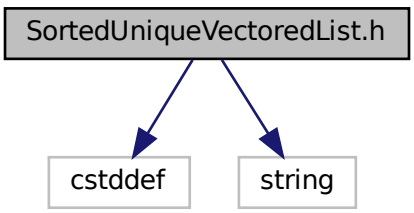
- struct SortedUniqueVectoredList::Bucket

6.5 SortedUniqueVectoredList.h File Reference

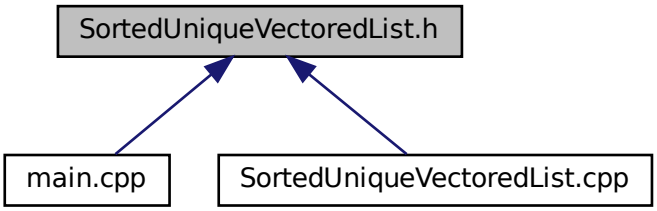
W ramach kolokwium probnego trzeba zaimplementowac mechanizmy kilku kontenerow w jednym: vector, list, deque, set, string. Szczegoly opisane sa w ramach metod klasy SortedUniqueVectoredList. Do ponizszych metod **sa testy** w pliku SortedUniqueVectoredListTests.cpp. **Jest to kolokwium probne, ktorego zrobienie samodzielne daje duze szanse na zdanie prostrzego, aczkolwiek podobnego, kolokwium wlasciwego za tydzien**

```
#include <cstddef>
#include <string>
```

Include dependency graph for SortedUniqueVectoredList.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SortedUniqueVectoredList

  Klasa SortedUniqueVectoredList, stanowiaca liste z glowa tablic dynamicznych trzymajacych teksty, ktore maja byc posortowane i unikalne. Bez obaw, nie musi byc optymalnie, schemat znajduje sie tutaj:

6.5.1 Detailed Description

W ramach kolokwium probnego trzeba zaimplementowac mechanizmy kilku kontenerow w jednym: vector, list, deque, set, string. Szczegoly opisane sa w ramach metod klasy SortedUniqueVectoredList. Do ponizszych metod **sa testy** w pliku SortedUniqueVectoredListTests.cpp. **Jest to kolokwium probne, ktorego zrobienie samodzielne daje duze szanse na zdanie prostrzego, aczkolwiek podobnego, kolokwium wlasciwego za tydzien**

Date

31 maja 2021

Index