

DOKUMENTACJA PRZEDSIĘWZIĘCIA PROGRAMISTYCZNEGO

DRZEWO GENEALOGICZNE

Członkowie Grupy „4z”:

Łukasz Janus - kierownik grupy

Bartosz Bukowski

Mateusz Marchelewicz

Łukasz Witek vel Witkowski

Łódź, 28 maja 2017

SPIS TREŚCI

1. RAPORT Z FAZY STRATEGICZNEJ	5
1.1. CEL PRZEDSIĘWZIĘCIA	5
1.2. ZAKRES PRZEDSIĘWZIĘCIA	5
1.3. OPIS SYSTEMÓW ZEWNĘTRZNYCH Z KTÓRYMI SYSTEM BĘDZIE WSPÓŁPRACOWAĆ	5
1.4. OGÓLNY OPIS WYMAGAŃ	5
1.5. OGÓLNY MODEL SYSTEMU	5
1.6. OPIS PROPONOWANEGO ROZWIĄZANIA	5
1.7. WSTĘPNY PROJEKT PROGRAMU	6
1.8. PROJEKT MENU, OPCJI EDYCJI DRZEWA ORAZ ZAPISU DANYCH OSOBY W DRZEWIE	6
1.9. OSZACOWANIE KOSZTÓW	7
1.10. WSTĘPNY HARMONOGRAM PRAC	7
1.11. PROPONOWANA WIZUALIZACJA PROGRAMU	8
2. RAPORT Z FAZY OKREŚLANIA WYMAGAŃ	9
2.1. WPROWADZENIE	9
2.2. OPIS PRZEWIDYWANEJ EWOLUCJI SYSTEMU	9
2.3. OPIS WYMAGAŃ FUNKCJONALNYCH	9
2.4. OPIS WYMAGAŃ NIEFUNKCJONALNYCH	10
2.5. OBSŁUGA WYJĄTKÓW	11
3. DOKUMENT Z FAZY ANALIZY	12
3.1. DIAGRAM PRZEPŁYWU DANYCH	14
3.2. DIAGRAM PRZYPADKÓW UŻYCIA	15
4. DOKUMENTACJA FAZY PROJEKTOWANIA	16
4.1. LISTA KLAS I POWIĄZANIA MIĘDZY KLASAMI	17
4.2. DIAGRAM POWIĄZAŃ MIĘDZY KLASAMI	19
5. DOKUMENTACJA FAZY IMPLEMENTACJI	20
6. RAPORT Z TESTÓW MODUŁÓW	49
7. RAPORT Z TESTÓW MODUŁÓW	57
8. DOKUMENTACJA ADMINISTRATORA	58
9. DOKUMENTACJA UŻYTKOWNIKA	62
9.1. INFORMACJE WSTĘPNE	62
9.2. PORUSZANIE SIĘ PO MENU	62
9.3. URUCHOMIENIE PROGRAMU	62
9.4. PORADY OGÓLNE	64
9.5. SCHEMAT DOSTĘPNYCH OPCJI	65
10. DOKUMENTACJA Z FAZY KONSERWACJI	66
11. RAPORT Z POSTĘPÓW PRODUKCJI OPROGRAMOWANIA (PROGRESS REPORT)	67
11.1. TABLICA POSTĘPÓW	67
12. DOKUMENTACJA QA (KONTROLA JAKOŚCI)	70
12.1. TABELA QA	70
12.2. ZGODNOŚĆ KODU ZE STANDARDAMI W FORMIE „QA CLASS DOCUMENT” ZAWIERAJĄCY KAŻDĄ KLASĘ I POTWIERDZAJĄCY JEJ ZGODNOŚĆ ZE STANDARDEM	71
13. DOKUMENTACJA TESTOWANIA	82
13.1. TABELA PROPONOWANYCH TESTÓW	82
13.2. RAPORT BŁĘDÓW	83

14.	INNE.....	84
-----	-----------	----

1. RAPORT Z FAZY STRATEGICZNEJ

1.1. *Cel przedsięwzięcia*

Celem zespołu jest stworzenie kompletnego programu pozwalającego na budowę drzewa genealogicznego.

1.2. *Zakres przedsięwzięcia*

Projekt ma na celu symulację średniego przedsięwzięcia.

1.3. *Opis systemów zewnętrznych z którymi system będzie współpracować*

Program będzie samodzielny – nie będzie korzystać z dodatkowych systemów/zasobów.

1.4. *Ogólny opis wymagań*

Program będzie umożliwiał stworzenie oraz edycję drzewa genealogicznego.

1.5. *Ogólny model systemu*

Każda osoba będzie miała przypisane ID. Na jego podstawie program, za pośrednictwem wyspecjalizowanych klas, będzie zapisywać i odczytywać informacje dotyczące każdej osoby w kilku plikach tekstowych (np. ID, imię, nazwisko, data urodzenia, śmierci). W innym pliku zapisywane będą informacje o osobie, a w innym o związkach lub relacjach.

1.6. *Opis proponowanego rozwiązania*

Z programu będzie mogła jednocześnie korzystać jedna osoba. Program nie będzie wymagał połączenia z Internetem. Opcjonalnie będzie można stworzone drzewo zapisać do pliku i wczytać go na innym komputerze.

Dana osoba stanowi pień, jej przodkowie w kolejnych pokoleniach będą stanowić coraz wyższe poziomy gałęzi. Przodkowie będą pokazywani bez rodzeństwa. Potomkowie danej osoby będą pokazywani w połączeniu ze współmałżonkiem. Jeśli dana osoba będzie posiadać dziecko nieślubne/adoptowane, będzie ono również wyświetlane w połączeniu z matką, jednak z zastrzeżeniem 'relacja inna'. Zakres kalendarzowy, od którego można dodać osobę, liczony będzie od roku zerowego ('0').

1.7. Wstępny projekt programu

W ramach struktury drzewa będzie możliwe:

- dodawanie osoby do drzewa
- edycja danych
- usuwanie (w przypadku błędnego dodania osoby)

Informacje dostępne do wprowadzenia członka drzewa lub jego edycji na każdym członku (wraz z poniższymi oznaczeniami):

- | | |
|------------------|------------------------------|
| - ID (#) | - rok (&) |
| - imię (\$) | - koniec pliku / rzędu () |
| - nazwisko (@) | - narodziny (%) |
| - płeć (!) | - ślub (*) |
| - dzień (^) | - zgon (+) |
| - miesiąc (~) | |

1.8. Projekt menu, opcji edycji drzewa oraz zapisu danych osoby w drzewie

Projekt dostępnych opcji dla menu głównego:

1. New Tree
 - Create New Tree
 - Import Tree
2. Load Tree
 - Display Tree
 - Edit Tree
 - Export Tree
 - Exit
3. Exit

Projekt dostępnych opcji dla menu edycji drzewa

<Nazwa drzewa>

1. Add a person
 - podaj imię, nazwisko, daty: narodziny, śmierć
 - dodaj datę ślubu (podmenu → dodaj osobę)
 - relacja inna: data (podmenu → dodaj osobę, → dodaj dzieci)
 - dodaj do (→ podaj imię i nazwisko) jako 'potomek'/'przodek'
2. Edit a person
 - podaj imię i nazwisko
3. Add a relation
4. Edit a relation
5. Exit.

Przykładowy zapis osoby w drzewie:

Jan Kowalski % 01.01.1900 + 02.02.1930	Związki: * 1919, Janina Nowak * 1921, Jadwiga Nowakowska
	Relacje inne: 1923 Anna Kowalewska (potomkowie)

1.9. *Oszacowanie kosztów*

???????????

1.10. *Wstępny harmonogram prac*

Praca planowana jest na okres marzec-maj 2017 roku. Wersja beta przedstawiona będzie ok. 4-5 czerwca, prezentacja 17 czerwca. Spotkania bezpośrednie zespołu odbywają się zgodnie z poniższym grafikiem:

- | | |
|------------------|------------------|
| 1) 11-12.03.2017 | 5) 22-23.04.2017 |
| 2) 18-19.03.2017 | 6) 06-07.05.2017 |
| 3) 01-02.04.2017 | 7) 13-14.05.2017 |
| 4) 08-09-04.2017 | 8) 27-28.05.2017 |

- raport oceny rozwiązań, zawierający informacje o rozważanych rozwiązaniach oraz przyczynach wyboru jednego z nich,
- opis wymaganych zasobów:
pracownicy – Łukasz Janus, Mateusz Marchelewicz, Łukasz Witek, Bartosz Bukowski
oprogramowanie – Visual Studio 2015 z wtyczką do GitHuba, LocMetrics
sprzęt – komputery z systemem Windows oraz zainstalowanym w/w Visual Studio 15 i programem do liczenia kodu LocMetrics
- definicje standardów.

Raporty wewnętrzne (dla firmy, wykonane dla potrzeb rozpatrywanego projektu),

- raport oceny rozwiązań, zawierający informacje o rozważanych rozwiązaniach oraz przyczynach wyboru jednego z nich,

- opis wymaganych zasobów — pracownicy, oprogramowanie, sprzęt,
- definicje (dodatkowych) standardów, np. ISO.
- harmonogram fazy analizy.
- diagram Gantta

1.11. Proponowana wizualizacja programu

a) aplikacja konsolowa (przykładowe zrzuty z menu)

```

C:\Windows\system32\cmd.exe

      |
      |
  =====
      |      |
  =====  =====
  | | | | | | | | | | | |
  | | | | | | | | | | | |

DRZEWO GENEALOGICZNE - FAMILY TREE

--> 1. New Tree [Create New Tree]
    2. Load Tree
    3. Exit

Use the arrows to navigate the menu ↑ ↓. Confirm your choice with ENTER.

```

```

C:\Windows\system32\cmd.exe

      |
      |
  =====
      |      |
  =====  =====
  | | | | | | | | | | | |
  | | | | | | | | | | | |

DRZEWO GENEALOGICZNE - FAMILY TREE

--> 1. Add a person
    [You can add a person to your tree]
    2. Edit a person
    3. Add a relation
    4. Edit a relation
    5. Exit

Use the arrows to navigate the menu ↑ ↓. Confirm your choice with ENTER.
Click SPACEBAR if you want back to main menu.

```


2. RAPORT Z FAZY OKREŚLANIA WYMAGAŃ

2.1. *Wprowadzenie*

Celem zespołu jest stworzenie kompletnego programu pozwalającego na budowę i edycję drzewa genealogicznego. Projekt ma na celu symulację średniego przedsięwzięcia.

2.2. *Opis przewidywanej ewolucji systemu*

W przyszłości planowany będzie interfejs użytkownika okienkowy (technologia WinAPI), oraz możliwość dodawania plików graficznych (zdjęć). Rozszerzenie danych personalnych i tym samym wyszukiwania osób po tych danych: miejsce urodzenia, miejsce śmierci, data rozvodu. Możliwość eksportu drzewa do pliku graficznego w celu wydruku.

2.3. *Opis wymagań funkcjonalnych*

- **możliwość tworzenia drzewa genealogicznego,**
- **możliwość podania nazwy tworzonego drzewa** (tworzone drzewo będzie można nazwać, nie będą mogły występować drzewa o tych samych nazwach),
- **możliwość zapisu tworzonego lub modyfikowanego drzewa** (program będzie umożliwiał zapis drzewa do pliku tekstowego oraz zapis postępu prac przy istniejącym drzewie),
- **możliwość importu/eksportu drzewa genealogicznego** (stworzone drzewo będzie można importować z/do pliku w celu jego przenoszenia między różnymi komputerami),
- **możliwość usunięcia istniejącego drzewa,**
- **program nie będzie miał możliwości działania z siecią Internet** (brak możliwości aktualizacji programu do nowszej wersji, brak możliwości przesyłania stworzonych drzew do innych osób za pośrednictwem sieci),
- **możliwość dodania osoby do istniejącego drzewa** (do stworzonego drzewa będzie można dołączać kolejne osoby),
- **dodawanie osoby** (możliwość podania imienia, tylko jednego nazwiska, płci, daty urodzenia, daty śmierci:
 - opcjonalnie, daty ślubu – opcjonalnie),
- **dodawanie kolejnych osób** (możliwość tworzenia podstawowych relacji: córka, syn, ojciec, matka, babcia, dziadek, brat, siostra),

- **możliwość dodania osoby do drzewa, jako potomka bądź przodka** (użytkownik będzie mógł wybrać jedną z tych opcji),
- **możliwość wyszukania danej osoby po nazwisku lub ID,**
- **możliwość wyświetlenia wybranego drzewa po nazwie** (można stworzyć kilka różnych drzew w ramach jednego programu),
- **możliwość wyświetlenia osoby z jej najbliższą rodziną** (np. mąż + żona, ich dzieci i dziadkowie),
- **możliwość edycji danych lub usunięcia wybranej osoby z drzewa,**
- **w przypadku nieznanymi danych program powinien uzupełnić pole treścią „dane nieznane”.**

2.4. Opis wymagań niefunkcjonalnych

Komputer klasy PC z zainstalowanym systemem MS Windows:

- procesor 1 GHz lub szybszy (x86 lub x64),
- 1 GB pamięci RAM (x86) lub 2 GB pamięci RAM (dla x64),
- 500 MB HDD,
- urządzenie graficzne z obsługą programu DirectX 9.

(Ostateczna wersja na końcu projektu!!)

np.:

Wydajność:

Liczba transakcji obsłużonych w ciągu sekundy

Czas odpowiedzi

Szybkość odświeżania ekranu

Rozmiar:

Wymagana pamięć RAM

Wymagana pamięć dyskowa

np.:

Łatwość użytkowania:

Czas niezbędny dla przeszkolenia użytkowników

Liczba stron dokumentacji

2.5. *Obsługa wyjątków*

1. Program powinien uniemożliwić dodania osoby o dacie ur. późniejszej od daty śmierci (np. Jan Nowak ur.12.12.2005, zm. 30.11.2000).
2. Program powinien uniemożliwić dodania matki o dacie ur. późniejszej od daty ur. dziecka (np. Marta Kowal ur.12.12.1981, ma córkę Olę ur. 20.03.1978).
3. Brak konfliktu przy zbieżności nazwisk (np. Marta Brzoza jako panna z domu Brzoza żeni się z Henrykiem Brzozą – wcześniej się nie znają mimo zbieżności nazwisk).
4. Program będzie umożliwiał wpisania tylko jednego nazwiska (np. mężczyzna i kobieta będą mogli mieć tylko JEDNO nazwisko – kobieta nazwisko z domu bądź po mężu).
5. Program powinien pozwolić dodać dwójkę dzieci z tym samym nazwiskiem i imieniem, ale TYLKO w przypadku, gdy jedno z dzieci umrze.

3. DOKUMENT Z FAZY ANALIZY

1. Opis stworzonego modelu:

- **funkcje systemu** (zostały uzgodnione z klientem),
- **wydajność systemu** (przeszukiwanie bazy 5000 pozycji w ciągu < 1 sek),
- **interfejsy zewnętrzne** (okienkowy),
- **wykonywane operacje** (dokładnie wyspecyfikowane),
- **wymagane zasoby sprzętowe** (określone),
- **sposoby weryfikacji** (weryfikacja = zgodność z wymaganiami),
- **sposoby testowania** (testowane są kolejne moduły w pliku *Main.cpp*),
- **sposoby dokumentowania** (projekt zawiera dwa dokumenty: firmy i przedsięwzięcia),
- **ochrona danych** (nie zachodzi),
- **przenośność** (przestrzegać, aby kod nie okienkowy działał pod linuxem/unixem, nie dotyczy),
- **jakość** (wymagana zgodność ze standardami firmy)
- **niezawodność** (nie dotyczy),
- **sposoby pielęgnacji** (kod może być łatwo modyfikowany/rozwijany),
- **bezpieczeństwo** (nie dotyczy),

2. Opis przypadków użycia:

- aktorzy,
 - a) użytkownik - u
 - b) (inny) komputer - ik
- podstawowy ciąg zdarzeń,
 - a) u: tworzy nowe drzewo genealogiczne
 - b) u: określa nazwę drzewa genealogicznego
 - c) u: dodaje osobę bazową
 - d) u: określa dane osoby bazowej (imię, nazwisko, płeć, datę urodzenia)
 - e) u: dodaje osobę powiązaną relacją do osoby bazowej
 - f) u: określa dane osoby powiązaną relacją
 - g) u: wyświetla zbudowane drzewo
 - h) u: wyszukuje/przegląda interesujące go osoby

- i) u: wyświetla dane interesujących go osób oraz powiązanych relacją osób
- j) u: zapisuje drzewo
- k) u: wczytuje drzewo i powtarza np. punkty c-j
- alternatywny ciąg zdarzeń,
 - a) u: użytkownik rezygnuje z drzewa i je usuwa
 - b) u: określa dodatkowe dane (np. datę śmierci, ślubu, rozvodu) lub usuwa osobę
 - c) u: określa dodatkowe dane (np. datę śmierci, ślubu, rozvodu) lub usuwa osobę
 - d) u: edytuje dane osoby
 - e) u: eksportuje plik na ik
 - f) u: po przed wczytaniem importuje plik z ik
- zależności czasowe,
 - g) wyświetlenie rozbudowanego drzewa może spowodować kilkusekundowe wczytywanie drzewa
 - j) zapis lub eksport do pliku może trwać kilka sekund
 - k) wczytanie lub import z pliku może trwać kilka sekund
- wartość uzyskana z przypadku użycia.
 - a) utworzenie w programie nazwanego drzewa
 - b) utworzenie w programie nowej osoby
 - c) utworzenie w programie nowej osoby
 - d) utworzenie pliku tekstowego z zapisanymi danymi
 - e) wygenerowanie drzewa oraz danych osób z pliku

3.1. Diagram przepływu danych

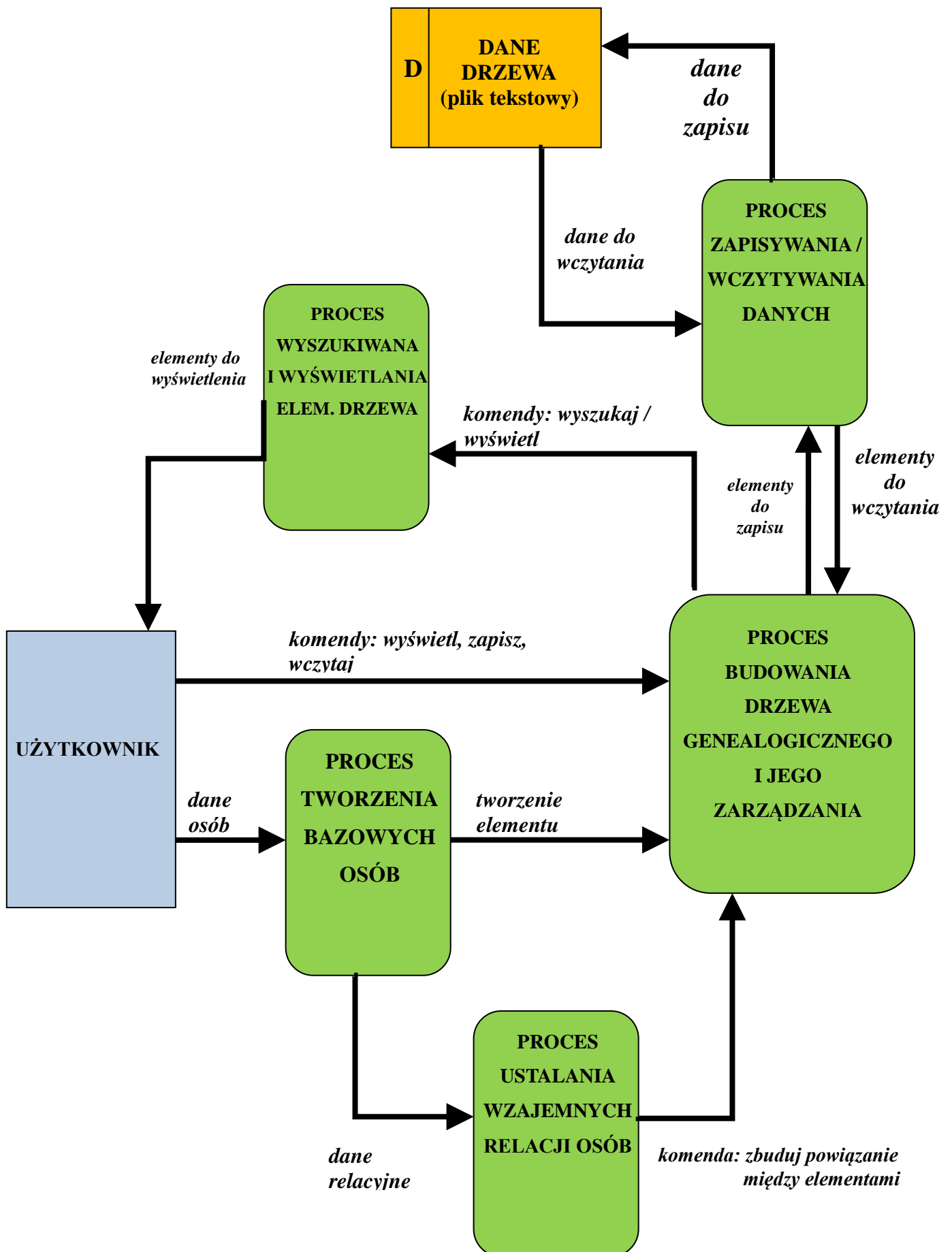


DIAGRAM PRZYPADKÓW UŻYCIA

4. DOKUMENTACJA FAZY PROJEKTOWANIA

Jeśli zaproponowano model obiektowy, to dokument powinien zawierać:

- wykaz klas,
- diagramy klas,
- diagramy przejść stanów,
- raporty:
 - definicji klas,
 - definicji pól,
 - definicji danych złożonych oraz elementarnych,
 - definicji metod
- diagramy UML,
- diagram działania programu.

4.1. *Lista klas i powiązania między klasami*

Klasa C_date	klasa potomna po C_day, C_month, C_year
Klasa C_day	klasa bazowa dla C_date
Klasa C_month	klasa bazowa dla C_date
Klasa C_year	klasa bazowa dla C_date

Klasa C_data	klasa podstawowa, bazowa dla C_first_name, C_last_name, C_gender, C_id
Klasa C_first_name	klasa pochodna, która dziedziczy po C_data
Klasa C_last_name	klasa pochodna, która dziedziczy po C_data
Klasa C_gender	klasa pochodna, która dziedziczy po C_data
Klasa C_id	klasa pochodna, która dziedziczy po C_data

Klasa C_relation	klasa podstawowa, bazowa dla C_children, C_parent, C_sibling
Klasa C_children	klasa pochodna, która dziedziczy po C_relation
Klasa C_parent	klasa pochodna, która dziedziczy po C_relation
Klasa C_partner	klasa pochodna, która dziedziczy po C_relation
Klasa C_grandparents	klasa pochodna, która dziedziczy po C_relation

Klasa C_grandchildren	klasa pochodna, która dziedziczy po C_relation
Klasa C_sibling	klasa pochodna, która dziedziczy po C_relation

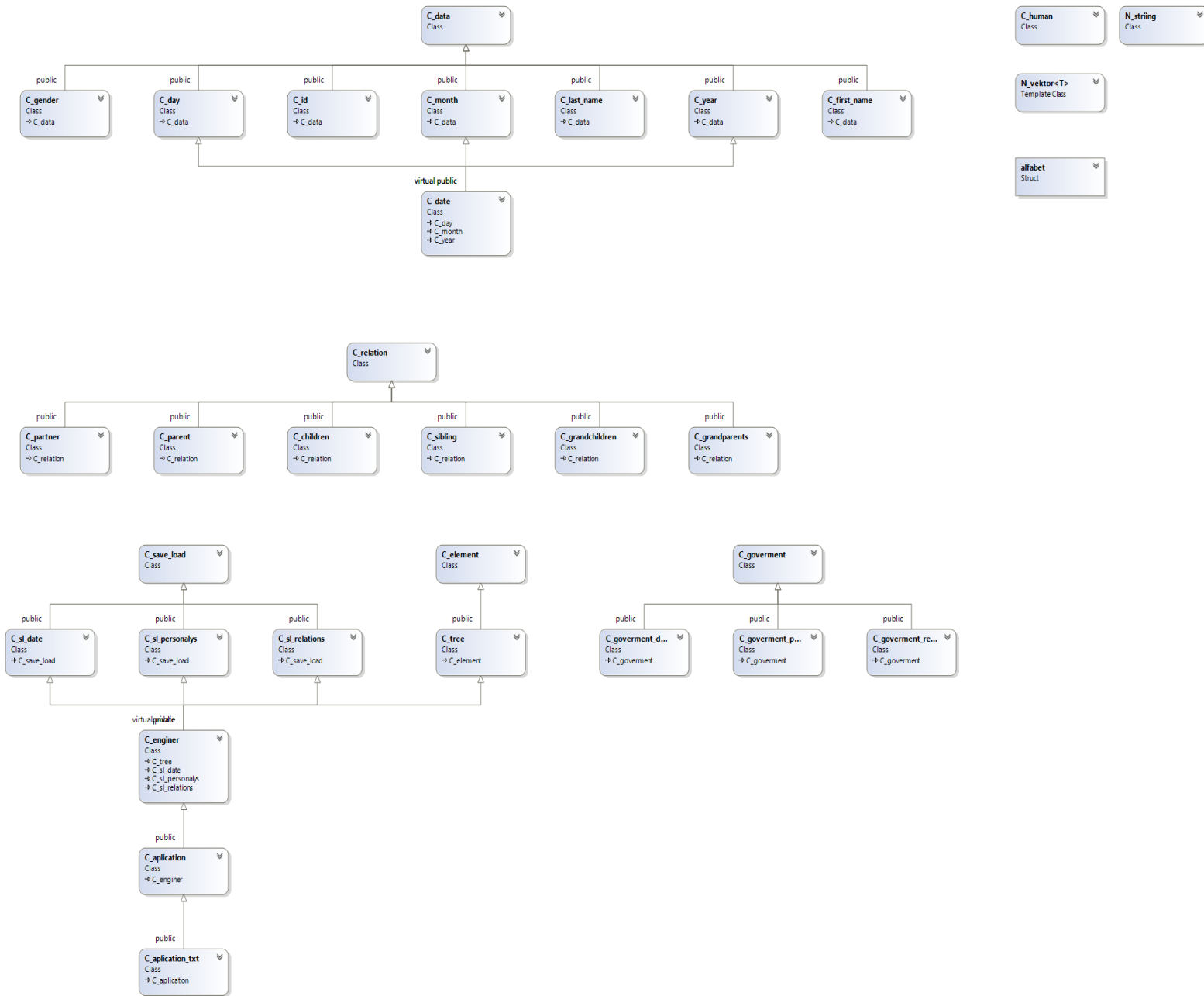
Klasa C_element - klasa podstawowa, bazowa dla C_tree, pochodna dla C_human,
Klasa C_tree - klasa podstawowa, dziedziczy i dzięki której działa C_engineer,
Klasa C_human – klasa podstawowa, bazowa dla C_first_name, C_last_name, C_id, C_date,
Klasa C_government - klasa bazowa dla C_government_date, C_government_personalys, C_government_relation,
Klasa C_government_date - klasa dziedzicząca po C_government,
Klasa C_government_personalys - klasa dziedzicząca po C_government ,
Klasa C_government_relation - klasa dziedzicząca po C_government

Klasa C_engineer - klasa podstawowa, bazowa dla C_tree, C_sl_date, C_sl_personalys,
Klasa C_save_load – klasa bazowa dla innych klas do wczytywania danych
Klasa C_sl_date – klasa dziedzicząca po C_save_load
Klasa C_sl_personalys – klasa dziedzicząca po C_save_load
Klasa C_sl_relations – klasa dziedzicząca po C_save_load

Klasa C_application - Klasa czysto abstrakcyjna dla klas poświęconych interfejsowi, klasa dziecko po klasie engineer
Klasa C_application_txt – klasa dziedzicząca po C_application

Klasa N_striing – klasa podstawowa, bazowa (zastępuje bibliotekę String)
Klasa N_vektor – klasa podstawowa, bazowa (zastępuje bibliotekę Vector)

4.2. Diagram powiązań między klasami

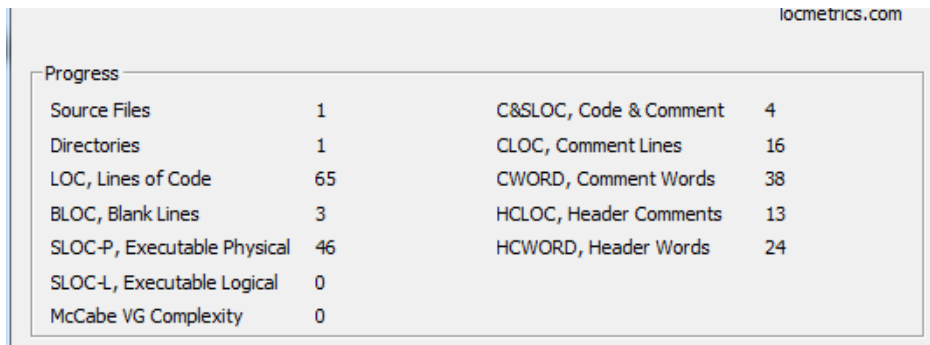


5. DOKUMENTACJA FAZY IMPLEMENTACJI

kod składający się z przetestowanych modułów

+ obliczone KLOC, McCabe'y, SLOC, KDSI dla każdego modułu (za pomocą programu!!)

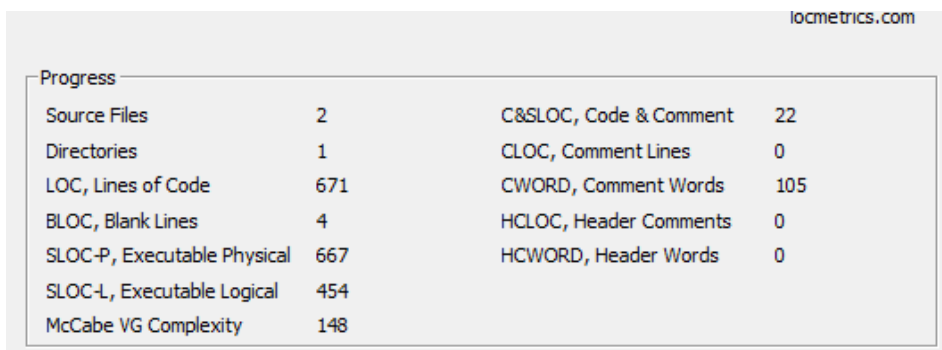
Folder Helpful (**definition.h**)



locmetrics.com

Progress			
Source Files	1	C&SLOC, Code & Comment	4
Directories	1	CLOC, Comment Lines	16
LOC, Lines of Code	65	CWORD, Comment Words	38
BLOC, Blank Lines	3	HCLOC, Header Comments	13
SLOC-P, Executable Physical	46	HCWORD, Header Words	24
SLOC-L, Executable Logical	0		
M McCabe VG Complexity	0		

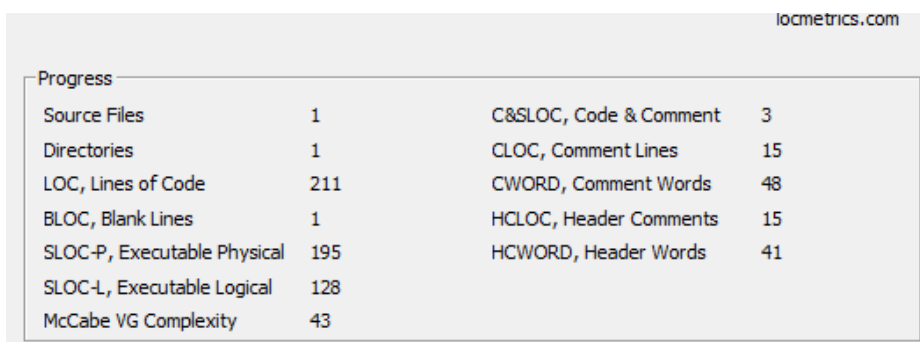
Folder Project Tools (**striing.h, striing.cpp**)



locmetrics.com

Progress			
Source Files	2	C&SLOC, Code & Comment	22
Directories	1	CLOC, Comment Lines	0
LOC, Lines of Code	671	CWORD, Comment Words	105
BLOC, Blank Lines	4	HCLOC, Header Comments	0
SLOC-P, Executable Physical	667	HCWORD, Header Words	0
SLOC-L, Executable Logical	454		
M McCabe VG Complexity	148		

Folder Project Tools (**Vektor.h**)



locmetrics.com

Progress			
Source Files	1	C&SLOC, Code & Comment	3
Directories	1	CLOC, Comment Lines	15
LOC, Lines of Code	211	CWORD, Comment Words	48
BLOC, Blank Lines	1	HCLOC, Header Comments	15
SLOC-P, Executable Physical	195	HCWORD, Header Words	41
SLOC-L, Executable Logical	128		
M McCabe VG Complexity	43		

.....

DOKUMENTACJA DOXYGEN

DRZEWO GENEALOGICZNE

Spis treści

Table of contents

Indeks hierarchiczny

Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

alfabet.....	25
C_data	28
C_day	30
C_date	29
C_first_name	32
C_first_name	32
C_gender	33
C_id	38
C_id	38
C_last_name	39
C_month	39
C_date	29
C_year	45
C_date	29
C_element	31
C_tree	44
C_goverment	34
C_goverment_date	34
C_goverment_personaly	35
C_goverment_relation	35
C_human	37
C_relation	41
C_children	27
C_grandchildren	36
C_grandparents	36
C_parent	40
C_partner	40
C_sibling	42
C_save_load	41
C_sl_date	42
C_enginer	32
C_aplication	25
C_aplication_txt	26

C_sl_personalys	43
C_enginer	32
C_sl_relations.....	44
C_enginer	32
N_striing.....	45
N_vektor< T >.....	47
N_vektor< C_children >.....	47
N_vektor< C_date >.....	47
N_vektor< C_goverment_date >.....	47
N_vektor< C_goverment_personaly >.....	47
N_vektor< C_goverment_relation >.....	47
N_vektor< C_grandchildren >.....	47
N_vektor< C_grandparents >.....	47
N_vektor< C_human >.....	47
N_vektor< C_last_name >.....	47
N_vektor< C_parent >.....	47
N_vektor< C_sibling >.....	47
N_vektor< C_tree >.....	47

Indeks klas

Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

alfabet	25
C_aplication	25
C_aplication_txt	26
C_children	27
C_data	28
C_date	29
C_day	30
C_element	31
C_enginer	32
C_first_name	32
C_gender	33
C_goverment	34
C_goverment_date	34
C_goverment_personaly	35
C_goverment_relation	35
C_grandchildren	36
C_grandparents	36
C_human	37
C_id	38
C_last_name	39
C_month	39
C_parent	40

C_partner	40
C_relation	41
C_save_load	41
C_sibling	42
C_sl_date	42
C_sl_personalys	43
C_sl_relations	44
C_tree	44
C_year	45
N_striing	45
N_vektor< T >	47

Dokumentacja klas

Dokumentacja struktury alfabet

Metody publiczne

1. bool **operator==** (alfabet &a)
2. bool **operator!=** (alfabet &a)
3. void **ladowanie_sz** (N_vektor< alfabet > &v)

Atrybuty publiczne

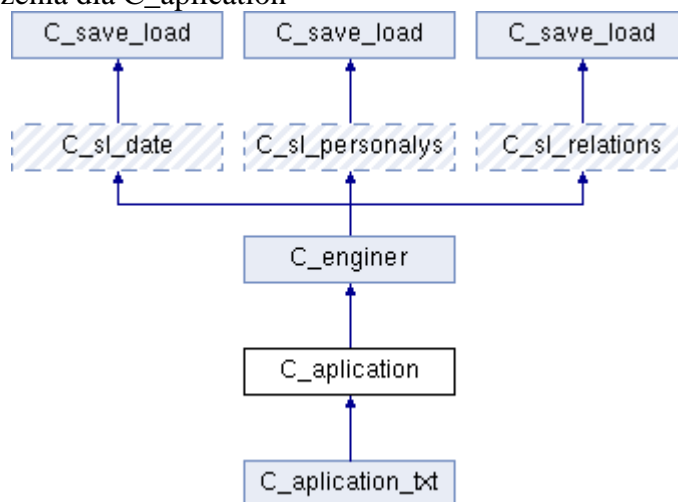
4. char **litera**
5. int **lp**
6. int **ascii**

Dokumentacja dla tej struktury została wygenerowana z pliku:

7. Data/Enginer/alphabet.h

Dokumentacja klasy C_application

Diagram dziedziczenia dla C_application



Metody publiczne

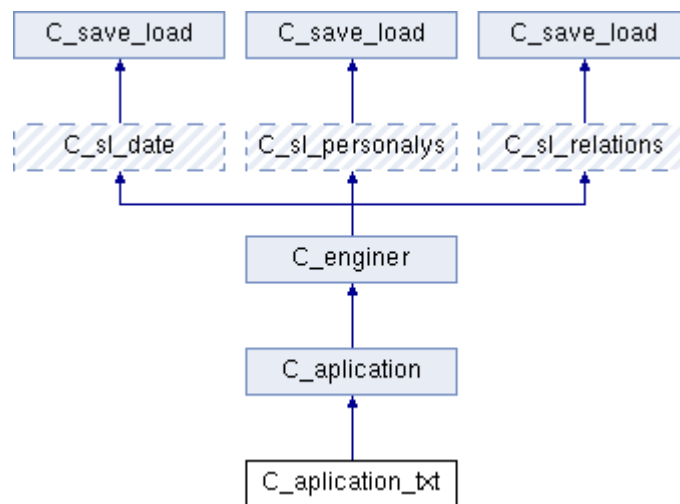
8. **C_application** (const **C_application** &application)
9. **C_application** & **operator=** (const **C_application** &application)
10. bool **operator==** (const **C_application** &application)
11. bool **operator!=** (const **C_application** &application)

Dokumentacja dla tej klasy została wygenerowana z plików:

12. Data/Interface/application.h
13. Data/Interface/application.cpp

Dokumentacja klasy C_application_txt

Diagram dziedziczenia dla C_application_txt



Metody publiczne

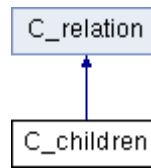
14. **C_application_txt** (const **C_application_txt** &application_txt)
15. **C_application_txt** & **operator=** (const **C_application_txt** &application_txt)
16. bool **operator==** (const **C_application_txt** &application_txt)
17. bool **operator!=** (const **C_application_txt** &application_txt)
18. void **SetWindow** (int Width, int Height)
19. void **MainMenu** ()
20. void **Sub1** ()
21. void **SubMenu2** ()
22. void **ImportTree** ()
23. void **EditTree** ()
24. void **DisplayTree** ()
25. void **SearchTree** ()
26. void **CreateLogo** ()

Dokumentacja dla tej klasy została wygenerowana z plików:

27. Data/Interface/application_txt.h
28. Data/Interface/application_txt.cpp

Dokumentacja klasy C_children

Diagram dziedziczenia dla C_children



Metody publiczne

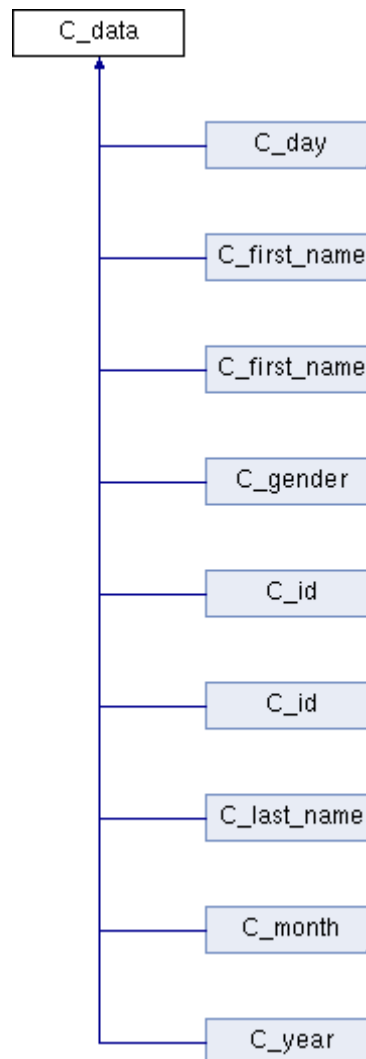
- 29. **C_children** (C_id &id)
- 30. **C_children** (const C_id &id)
- 31. **C_children** (const C_children &children)
- 32. **C_children & operator=** (const C_children &children)
- 33. **bool operator==** (const C_children &children)
- 34. **bool operator!=** (const C_children &children)
- 35. **virtual void m_get_id** (C_id &id)
- 36. **virtual C_id m_set_id** ()
- 37. **virtual int m_set_variable** ()
- 38. **virtual void m_get_complete_content** (N_striing data)
- 39. **virtual void m_get_complete_content** (C_id index, C_id value)

Dokumentacja dla tej klasy została wygenerowana z plików:

- 40. Data/Relation/children.h
- 41. Data/Relation/children.cpp

Dokumentacja klasy C_data

Diagram dziedziczenia dla C_data



Metody publiczne

- 42. **C_data** (N_striing string)
- 43. **C_data** (const **C_data** &data)
- 44. **C_data** & **operator=** (const **C_data** &data)
- 45. bool **operator==** (const **C_data** &data)
- 46. bool **operator!=** (const **C_data** &data)
- 47. virtual bool **m_wchat_is** ()=0
- 48. virtual void **m_get_contens** (N_striing &contens)=0
- 49. virtual N_striing **m_set_contens** ()=0
- 50. virtual int **m_set_variable** ()=0
- 51. N_striing **m_what_type** ()
- 52. **C_data** (N_striing string)
- 53. **C_data** (const **C_data** &data)
- 54. **C_data** & **operator=** (const **C_data** &data)
- 55. bool **operator==** (const **C_data** &data)

56. **bool operator!=** (const **C_date** &data)

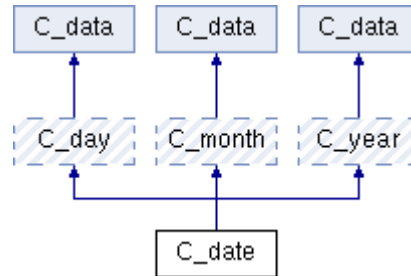
Dokumentacja dla tej klasy została wygenerowana z plików:

57. Data/Personalny/data.h

58. Data/Personalny/data.cpp

Dokumentacja klasy C_date

Diagram dziedziczenia dla C_date



Metody publiczne

- 59. **C_date** (char value)
- 60. **C_date** (const **C_date** &d)
- 61. **C_date & operator=** (const **C_date** &d)
- 62. **bool operator==** (const **C_date** &d)
- 63. **bool operator!=** (const **C_date** &d)
- 64. **C_day m_set_day** ()
- 65. **C_month m_set_month** ()
- 66. **C_year m_set_year** ()
- 67. **N_striing m_set_DD_MM_YYYY** ()
- 68. **N_striing m_set_MM_DD_YYYY** ()
- 69. **N_striing m_set_YYYY_MM_DD** ()
- 70. **N_striing m_set_YYYY_DD_MM** ()
- 71. **void m_get_DD_MM_YYYY** (**C_day** &day, **C_month** &month, **C_year** &year)
- 72. **void m_shift_day** (**C_day** day)
- 73. **void m_shift_day** (int day)
- 74. **void m_shift_day** (**N_striing** day)
- 75. **void m_shift_month** (**C_month** month)
- 76. **void m_shift_month** (int month)
- 77. **void m_shift_month** (**N_striing** month)
- 78. **void m_shift_year** (**C_year** year)
- 79. **void m_shift_year** (int year)
- 80. **void m_shift_year** (**N_striing** year)
- 81. **void m_clear** ()
- 82. **N_striing m_what_type_date** ()
- 83. **void m_shift_char** (char value)
- 84. **void m_get_type** (**N_striing** value)

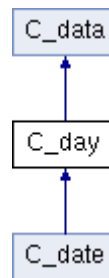
Dodatkowe Dziedziczone Składowe

Dokumentacja dla tej klasy została wygenerowana z plików:

- 85. Data/Date/date.h
- 86. Data/Date/date.cpp

Dokumentacja klasy C_day

Diagram dziedziczenia dla C_day



Metody publiczne

- 87. C_day (N_striing &day)
- 88. C_day (int day)
- 89. C_day (const C_day &C)
- 90. C_day & operator= (const C_day &C)
- 91. bool operator== (const C_day &C)
- 92. bool operator!= (const C_day &C)
- 93. virtual bool m_wchat_is ()
- 94. virtual void m_get_contens (N_striing &contens)
- 95. virtual int m_set_variable ()
- 96. N_striing m_day_set ()
- 97. void m_get_day (N_striing &contens)

Metody chronione

- 98. int m_set_value_day ()
- 99. void m_get_value_day (int value)

Atrybuty chronione

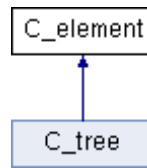
- 100.int i_data_day = NULL

Dokumentacja dla tej klasy została wygenerowana z plików:

- 101.Data/Date/day.h
- 102.Data/Date/day.cpp

Dokumentacja klasy C_element

Diagram dziedziczenia dla C_element



Metody publiczne

```
103.C_element (const C_human &human)
104.C_element (const C_element &human)
105.C_element & operator= (const C_element &human)
106.bool operator== (const C_element &human)
107.bool operator!= (const C_element &human)
108.void m_get_children (C_children &children)
109.void m_get_parent (C_parent &parent)
110.void m_get_sibling (C_sibling &sibling)
111.void m_get_grandchildren (C_grandchildren &grandchildren)
112.void m_get_grandparents (C_grandparents &grandparents)
113.void m_update_children (int value, C_children &children)
114.void m_update_parent (int value, C_parent &parent)
115.void m_update_sibling (int value, C_sibling &sibling)
116.void m_update_human (const C_human &human)
117.void m_update_grandchildren (int value, C_grandchildren &human)
118.void m_update_grandparents (int value, C_grandparents &human)
119.C_human m_set_Human ()
120.C_children m_set_children ()
121.C_parent m_set_parent ()
122.C_sibling m_set_sibling ()
123.C_children m_set_children (int value)
124.C_parent m_set_parent (int value)
125.C_sibling m_set_sibling (int value)
126.C_grandchildren m_set_grandchildren (int value)
127.C_grandparents m_set_grandparents (int value)
128.C_element & m_clean ()
129.void m_clean_children ()
130.void m_clean_parent ()
131.void m_clean_sibling ()
132.void m_clean_grandparents ()
133.void m_clean_grandchildren ()
134.void m_delete_children ()
135.void m_delete_parent ()
136.void m_delete_sibling ()
137.void m_delete_children (int value)
138.void m_delete_parent (int value)
139.void m_delete_sibling (int value)
140.void m_delete_grandchildren (int value)
141.void m_delete_grandparents (int value)
```

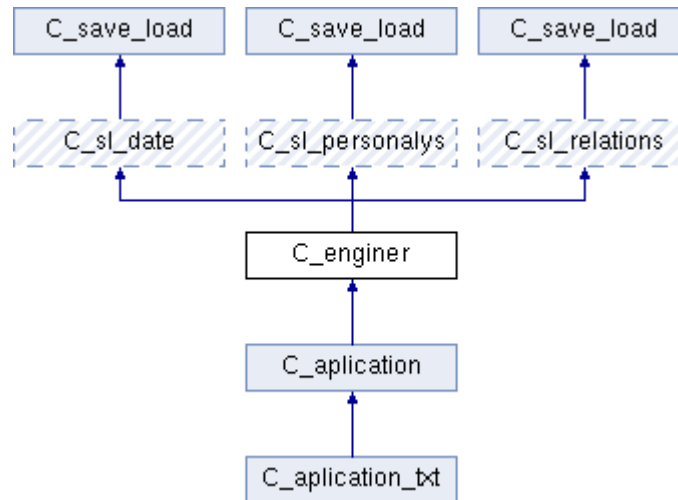
Dokumentacja dla tej klasy została wygenerowana z plików:

142.Data/Databases/element.h

143.Data/Databases/element.cpp

Dokumentacja klasy C_engineer

Diagram dziedziczenia dla C_engineer



Metody publiczne

- 144.C_engineer (const C_engineer &engineer)
- 145.C_engineer & operator= (const C_engineer &engineer)
- 146.bool operator== (const C_engineer &engineer)
- 147.bool operator!= (const C_engineer &engineer)
- 148.int m_set_index ()
- 149.void m_load_files ()
- 150.void m_save_files ()
- 151.void m_new_human (C_human human)
- 152.C_human & m_create_human (C_id id_finter)

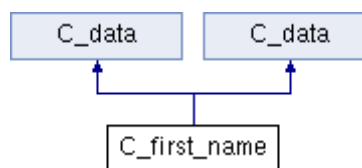
Dokumentacja dla tej klasy została wygenerowana z plików:

153.Data/Enginer/engineer.h

154.Data/Enginer/engineer.cpp

Dokumentacja klasy C_first_name

Diagram dziedziczenia dla C_first_name



Metody publiczne

```
155.C_first_name (N_striing &first)
156.C_first_name (const C_first_name &first)
157.C_first_name & operator= (const C_first_name &first)
158.bool operator== (const C_first_name &first)
159.bool operator!= (const C_first_name &first)
160.bool operator> (C_first_name &first)
161.bool operator< (C_first_name &first)
162.virtual bool m_wchat_is ()
163.virtual void m_get_contens (N_striing &contens)
164.virtual N_striing m_set_contens ()
165.virtual int m_set_variable ()
166.C_first_name (N_striing &first)
167.C_first_name (const C_first_name &first)
168.C_first_name & operator= (const C_first_name &first)
169.bool operator== (const C_first_name &first)
170.bool operator!= (const C_first_name &first)
```

Przyjaciele

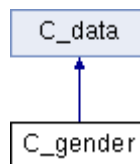
```
171.std::ostream & operator<< (std::ostream &is, C_first_name &first)
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
172.Data/Personal/first_name.h
173.Data/Personal/first_name.cpp
```

Dokumentacja klasy C_gender

Diagram dziedziczenia dla C_gender



Metody publiczne

```
174.C_gender (bool gender)
175.C_gender (N_striing &gender)
176.C_gender (const C_gender &C)
177.C_gender & operator= (const C_gender &C)
178.bool operator== (const C_gender &C)
179.bool operator!= (const C_gender &C)
180.virtual bool m_wchat_is ()
181.virtual void m_get_contens (N_striing &contens)
182.virtual N_striing m_set_contens ()
183.virtual int m_set_variable ()
```

Przyjaciele

184.std::ostream & **operator**<< (std::ostream &is, **C_gender** &gender)

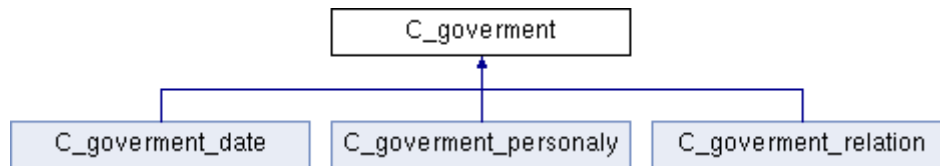
Dokumentacja dla tej klasy została wygenerowana z plików:

185.Data/Personal/gender.h

186.Data/Personal/gender.cpp

*Dokumentacja klasy **C_government***

Diagram dziedziczenia dla **C_government**



Metody publiczne

187.**C_government** (const **C_government** &government)

188.**C_government** & **operator**= (const **C_government** &government)

189.bool **operator**== (const **C_government** &government)

190.bool **operator**!= (const **C_government** &government)

191.virtual bool **m_wchat_is** ()=0

192.virtual void **m_get_contens** (**N_striing** &contens)=0

193.virtual **N_striing** **m_set_contens** ()=0

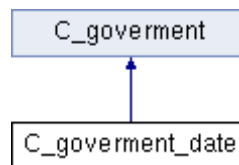
Dokumentacja dla tej klasy została wygenerowana z plików:

194.Data/Databases/government.h

195.Data/Databases/government.cpp

*Dokumentacja klasy **C_government_date***

Diagram dziedziczenia dla **C_government_date**



Metody publiczne

196.**C_government_date** (const **C_government_date** &government_date)

197.**C_government_date** & **operator**= (const **C_government_date** &government_date)

198.bool **operator**== (const **C_government_date** &government_date)

199.bool **operator**!= (const **C_government_date** &government_date)

200.virtual bool **m_wchat_is** ()

201.virtual void **m_get_contens** (**N_striing** &contens)

202.virtual **N_striing** **m_set_contens** ()

203.int **m_set_value_id** ()

204.**C_day** **m_set_value_day** ()

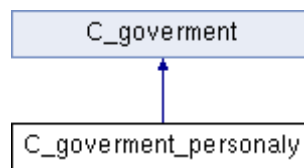
```
205.C_month m_set_value_month ()
206.C_year m_set_value_year ()
207.N_vektor< C_date > m_set_value_V_date ()
208.N_vektor< C_day > m_set_value_V_day ()
209.N_vektor< C_month > m_set_value_V_month ()
210.N_vektor< C_year > m_set_value_V_year ()
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
211.Data/Databases/goverment_date.h
212.Data/Databases/goverment_date.cpp
```

Dokumentacja klasy C_goverment_personaly

Diagram dziedziczenia dla C_goverment_personaly



Metody publiczne

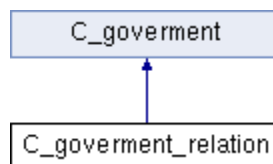
```
213.C_goverment_personaly (const C_goverment_personaly &goverment_personaly)
214.C_goverment_personaly & operator= (const C_goverment_personaly &goverment_personaly)
215.bool operator== (const C_goverment_personaly &goverment_personaly)
216.bool operator!= (const C_goverment_personaly &goverment_personaly)
217.virtual bool m_wchat_is ()
218.virtual void m_get_contens (N_striing &contens)
219.virtual N_striing m_set_contens ()
220.int m_set_value_id ()
221.C_first_name m_set_value_first_name ()
222.N_vektor< C_last_name > m_set_value_last_name ()
223.C_gender m_set_value_gender ()
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
224.Data/Databases/goverment_personaly.h
225.Data/Databases/goverment_personaly.cpp
```

Dokumentacja klasy C_goverment_relation

Diagram dziedziczenia dla C_goverment_relation



Metody publiczne

```
226.C_goverment_relation (const C_goverment_relation &goverment_relation)
227.C_goverment_relation & operator= (const C_goverment_relation &goverment_relation)
228.bool operator== (const C_goverment_relation &goverment_relation)
```

```

229.bool operator!= (const C_government_relation &government_relation)
230.virtual bool m_wchat_is ()
231.virtual void m_get_contens (N_striing &contens)
232.virtual N_striing m_set_contens ()
233.int m_set_value_id ()
234.N_vektor< C_children > m_set_value_children ()
235.N_vektor< C_parent > m_set_value_parent ()
236.N_vektor< C_grandchildren > m_set_value_grandchildren ()
237.N_vektor< C_grandparents > m_set_value_grandparents ()
238.N_vektor< C_sibling > m_set_value_sibling ()

```

Dokumentacja dla tej klasy została wygenerowana z plików:

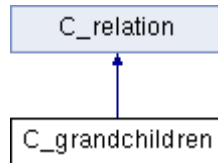
```

239.Data/Databases/government_relation.h
240.Data/Databases/government_relation.cpp

```

Dokumentacja klasy C_grandchildren

Diagram dziedziczenia dla C_grandchildren



Metody publiczne

```

241.C_grandchildren (const C_grandchildren &grandchildren)
242.C_grandchildren & operator= (const C_grandchildren &grandchildren)
243.bool operator== (const C_grandchildren &grandchildren)
244.bool operator!= (const C_grandchildren &grandchildren)
245.virtual void m_get_id (C_id &id)
246.virtual C_id m_set_id ()
247.virtual int m_set_variable ()
248.virtual void m_get_complete_content (N_striing data)
249.virtual void m_get_complete_content (C_id index, C_id value)

```

Dokumentacja dla tej klasy została wygenerowana z plików:

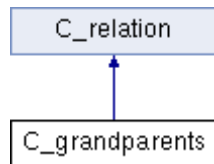
```

250.Data/Relation/grandchildren.h
251.Data/Relation/grandchildren.cpp

```

Dokumentacja klasy C_grandparents

Diagram dziedziczenia dla C_grandparents



Metody publiczne

252. **C_grandparents** (const **C_grandparents** &grandparents)
253. **C_grandparents** & **operator=** (const **C_grandparents** &grandparents)
254. **bool operator==** (const **C_grandparents** &grandparents)
255. **bool operator!=** (const **C_grandparents** &grandparents)
256. **virtual void m_get_id** (**C_id** &id)
257. **virtual C_id m_set_id** ()
258. **virtual int m_set_variable** ()
259. **virtual void m_get_complete_content** (**N_striing** data)
260. **virtual void m_get_complete_content** (**C_id** index, **C_id** value)

Dokumentacja dla tej klasy została wygenerowana z plików:

261. Data/Relation/grandparents.h
262. Data/Relation/grandparents.cpp

Dokumentacja klasy C_human

Metody publiczne

263. **C_human** (**C_id** &id)
264. **C_human** (const **C_human** &human)
265. **C_human** & **operator=** (const **C_human** &human)
266. **bool operator==** (const **C_human** &human)
267. **bool operator!=** (const **C_human** &human)
268. **void m_get_first_name** (**C_first_name** &f_name)
269. **void m_get_first_name** (**N_striing** &f_name)
270. **void m_get_last_name** (**C_last_name** &l_name)
271. **void m_get_last_name** (**N_striing** &l_name)
272. **void m_get_gender** (**C_gender** &gender)
273. **void m_get_gender** (**N_striing** &gender)
274. **void m_get_gender** (**bool** gender)
275. **void m_shift_id** (**N_striing** &id)
276. **void m_shift_id** (**int** id)
277. **void m_shift_id** (**C_id** &id)
278. **void m_get_date** (**C_date** date)
279. **void m_delete_first_name** ()
280. **void m_delete_last_name** (**int** value)
281. **void m_delete_last_name** ()
282. **void m_delete_gender** ()
283. **void m_delete_date** (**int** value)
284. **void m_delete_date** ()
285. **void m_update_date** (**int** value, **C_date** &date)
286. **void m_update_last_name** (**int** value, **C_last_name** &l_name)
287. **void m_update_last_name** (**int** value, **N_striing** &l_name)
288. **C_human** & **m_clear** ()
289. **C_human** & **m_clear_date** ()
290. **C_human** & **m_clear_last_name** ()
291. **C_first_name** **m_set_first_name** ()
292. **C_last_name** **m_set_last_name** ()
293. **C_last_name** **m_set_last_name** (**int** value)
294. **C_gender** **m_set_gender** ()

```

295.C_id m_set_id ()
296.C_date m_set_date (int value)
297.C_date m_set_date ()
298.N_vektor< C_date > m_set_Vdate ()
299.N_vektor< C_last_name > m_set_V_last_name ()

```

Przyjaciele

```

300.std::ostream & operator<< (std::ostream &is, C_human &human)

```

Dokumentacja dla tej klasy została wygenerowana z plików:

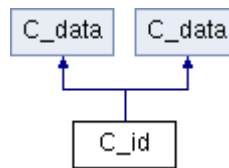
```

301.Data/Databases/human.h
302.Data/Databases/human.cpp

```

Dokumentacja klasy C_id

Diagram dziedziczenia dla C_id



Metody publiczne

```

303.C_id (char *id)
304.C_id (N_striing &id, bool t)
305.C_id (int id)
306.C_id (const C_id &C)
307.C_id & operator= (const C_id &C)
308.bool operator== (const C_id &C)
309.bool operator!= (const C_id &C)
310.bool operator> (C_id &id)
311.bool operator< (C_id &id)
312.virtual bool m_wchat_is ()
313.virtual void m_get_contens (N_striing &contens)
314.virtual N_striing m_set_contens ()
315.virtual int m_set_variable ()
316.void m_get_contens (int value)
317.C_id (const C_id &C)
318.C_id & operator= (const C_id &C)
319.bool operator== (const C_id &C)
320.bool operator!= (const C_id &C)

```

Dokumentacja dla tej klasy została wygenerowana z plików:

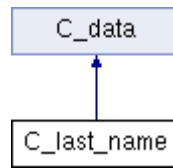
```

321.Data/Personal/id.h
322.Data/Personal/id.cpp

```

Dokumentacja klasy C_last_name

Diagram dziedziczenia dla C_last_name



Metody publiczne

```
323.C_last_name (N_striing &last)
324.C_last_name (const C_last_name &last)
325.C_last_name & operator= (const C_last_name &last)
326.bool operator== (const C_last_name &last)
327.bool operator!= (const C_last_name &last)
328.bool operator< (C_last_name &last)
329.bool operator> (C_last_name &last)
330.virtual bool m_wchat_is ()
331.virtual void m_get_contens (N_striing &contens)
332.virtual N_striing m_set_contens ()
333.virtual int m_set_variable ()
```

Przyjaciele

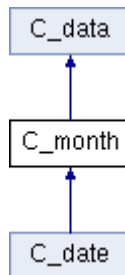
```
334.std::ostream & operator<< (std::ostream &is, C_last_name &last)
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
335.Data/Personal/last_name.h
336.Data/Personal/last_name.cpp
```

Dokumentacja klasy C_month

Diagram dziedziczenia dla C_month



Metody publiczne

```
337.C_month (N_striing &month)
338.C_month (int month)
339.C_month (const C_month &C)
340.C_month & operator= (const C_month &C)
341.bool operator== (const C_month &C)
342.bool operator!= (const C_month &C)
343.virtual bool m_wchat_is ()
```

```
344.virtual void m_get_contens (N_striing &contens)
345.virtual int m_set_variable ()
346.N_striing m_month_set ()
347.void m_get_month (N_striing &contens)
```

Metody chronione

```
348.int m_set_value_month ()
349.void m_get_value_month (int value)
```

Atrybuty chronione

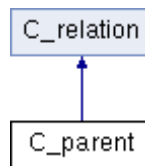
```
350.int i_data_month
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
351.Data/Date/month.h
352.Data/Date/month.cpp
```

Dokumentacja klasy C_parent

Diagram dziedziczenia dla C_parent



Metody publiczne

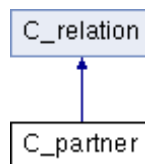
```
353.C_parent (C_id &id)
354.C_parent (const C_id &id)
355.C_parent (const C_parent &parent)
356.C_parent & operator= (const C_parent &parent)
357.bool operator== (const C_parent &parent)
358.bool operator!= (const C_parent &parent)
359.virtual void m_get_id (C_id &id)
360.virtual C_id m_set_id ()
361.virtual int m_set_variable ()
362.virtual void m_get_complete_content (N_striing data)
363.virtual void m_get_complete_content (C_id index, C_id value)
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
364.Data/Relation/parent.h
365.Data/Relation/parents.cpp
```

Dokumentacja klasy C_partner

Diagram dziedziczenia dla C_partner



Metody publiczne

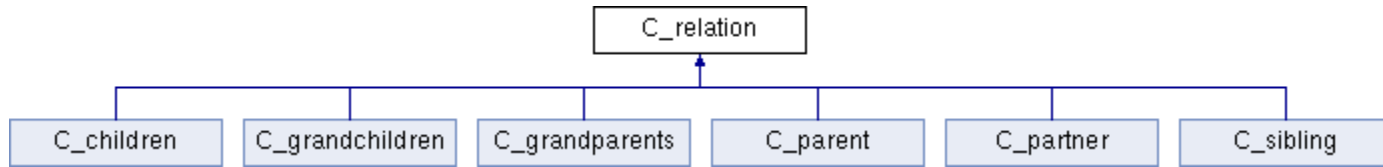
```
366.C_partner (const C_partner &partner)
367.C_partner & operator= (const C_partner &partner)
368.bool operator== (const C_partner &partner)
369.bool operator!= (const C_partner &partner)
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
370.Data/Relation/partner.h
371.Data/Relation/partner.cpp
```

Dokumentacja klasy C_relation

Diagram dziedziczenia dla C_relation



Metody publiczne

```
372.C_relation (N_string reat)
373.C_relation (const C_relation &relat)
374.C_relation & operator= (const C_relation &relat)
375.bool operator== (const C_relation &relat)
376.bool operator!= (const C_relation &relat)
377.virtual void m_get_id (C_id &id)=0
378.virtual C_id m_set_id ()=0
379.N_string m_what_type ()
380.virtual int m_set_variable ()=0
381.virtual void m_get_complete_content (N_string data)=0
382.virtual void m_get_complete_content (C_id index, C_id value)=0
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
383.Data/Relation/relation.h
384.Data/Relation/relation.cpp
```

Dokumentacja klasy C_save_load

Diagram dziedziczenia dla C_save_load

Metody publiczne

```
385.C_save_load (const C_save_load &save_load)
386.C_save_load & operator= (const C_save_load &save_load)
387.bool operator== (const C_save_load &save_load)
388.bool operator!= (const C_save_load &save_load)
389.virtual N_string m_cypher_on (N_string data)=0
390.virtual N_string m_cypher_off (N_string data)=0
```

Dokumentacja dla tej klasy została wygenerowana z plików:

391.Data/Enginer/save_load.h
392.Data/Enginer/save_load.cpp

392.1. *Dokumentacja klasy C_sibling*

Diagram dziedziczenia dla C_sibling

Metody publiczne

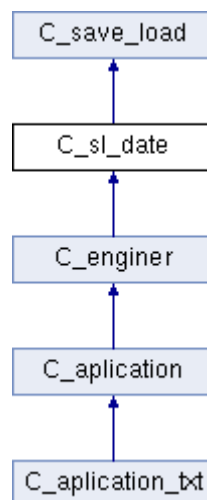
393.C_sibling (C_id &id)
394.C_sibling (const C_id &id)
395.C_sibling (const C_sibling &sib)
396.C_sibling & operator= (const C_sibling &sib)
397.bool operator== (const C_sibling &sib)
398.bool operator!= (const C_sibling &sib)
399.virtual void m_get_id (C_id &id)
400.virtual C_id m_set_id ()
401.virtual int m_set_variable ()
402.virtual void m_get_complete_content (N_striing data)
403.virtual void m_get_complete_content (C_id index, C_id value)

Dokumentacja dla tej klasy została wygenerowana z plików:

404.Data/Relation/sibling.h
405.Data/Relation/sibling.cpp

Dokumentacja klasy C_sl_date

Diagram dziedziczenia dla C_sl_date



Metody publiczne

406.C_sl_date (const C_sl_date &sl_date)
407.C_sl_date & operator= (const C_sl_date &sl_date)
408.bool operator== (const C_sl_date &sl_date)
409.bool operator!= (const C_sl_date &sl_date)

```

410.void m_file_date (bool what)
411.virtual N_string m_cypher_on (N_string data)
412.virtual N_string m_cypher_off (N_string data)
413.void m_get_new_date (C_id id, N_vektor< C_date > V_date)
414.C_goverment_date & operator[] (int i)
415.C_goverment_date m_set_gover_date (int i)
416.N_vektor< C_date > m_set_V_date (int i)

```

Dokumentacja dla tej klasy została wygenerowana z plików:

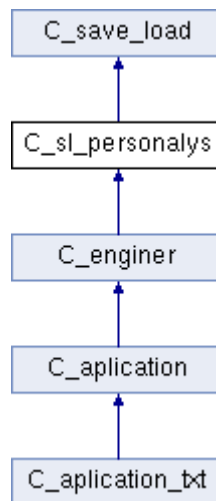
```

417.Data/Enginer/sl_date.h
418.Data/Enginer/sl_date.cpp

```

Dokumentacja klasy C_sl_personalys

Diagram dziedziczenia dla C_sl_personalys



Metody publiczne

```

419.C_sl_personalys (const C_sl_personalys &sl_personalys)
420.C_sl_personalys & operator= (const C_sl_personalys &sl_personalys)
421.bool operator== (const C_sl_personalys &sl_personalys)
422.bool operator!= (const C_sl_personalys &sl_personalys)
423.void m_load_file_personalys (bool what)
424.virtual N_string m_cypher_on (N_string data)
425.virtual N_string m_cypher_off (N_string data)
426.void m_add_new_personalys (C_id id, C_first_name first, N_vektor< C_last_name > Last,
    C_gender gender)
427.C_goverment_personalys & operator[] (int i)
428.C_goverment_personalys m_set_gover_personalys (int i)

```

Dokumentacja dla tej klasy została wygenerowana z plików:

```

429.Data/Enginer/sl_personalys.h
430.Data/Enginer/sl_personalys.cpp

```

Dokumentacja klasy C_sl_relations

Diagram dziedziczenia dla C_sl_relations

Metody publiczne

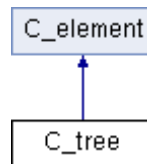
```
431.C_sl_relations (const C_sl_relations &sl_relations)
432.C_sl_relations & operator= (const C_sl_relations &sl_relations)
433.bool operator== (const C_sl_relations &sl_relations)
434.bool operator!= (const C_sl_relations &sl_relations)
435.void m_load_file_relation (bool what)
436.virtual N_striing m_cypher_on (N_striing data)
437.virtual N_striing m_cypher_off (N_striing data)
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
438.Data/Enginer/sl_relations.h
439.Data/Enginer/sl_relations.cpp
```

Dokumentacja klasy C_tree

Diagram dziedziczenia dla C_tree



Metody publiczne

```
440.C_tree (const C_human &human)
441.C_tree (const C_tree &tree)
442.C_tree & operator= (const C_tree &tree)
443.bool operator== (const C_tree &tree)
444.bool operator!= (const C_tree &tree)
445.void m_get_human (C_human &human)
446.void m_update_human (int value, C_human &human)
447.void m_delete_human (int value)
448.C_human m_set_human ()
449.C_human m_set_human (int value)
450.C_human m_set_human_index ()
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
451.Data/Databases/tree.h
452.Data/Databases/tree.cpp
```

452.1. *Dokumentacja klasy C_year*

Diagram dziedziczenia dla C_year

Metody publiczne

```
453.C_year (N_striing &year)
454.C_year (int year)
455.C_year (const C_year &C)
456.C_year & operator= (const C_year &C)
457.bool operator== (const C_year &C)
458.bool operator!= (const C_year &C)
459.virtual bool m_wchat_is ()
460.virtual void m_get_contens (N_striing &contens)
461.virtual int m_set_variable ()
462.void m_get_year (N_striing &contens)
463.N_striing m_year_set ()
```

Metody chronione

```
464.int m_set_value_year ()
465.void m_get_value_year (int value)
```

Atrybuty chronione

```
466.int i_data_year
```

Dokumentacja dla tej klasy została wygenerowana z plików:

```
467.Data/Date/year.h
468.Data/Date/year.cpp
```

Dokumentacja klasy N_striing

Metody publiczne

```
469.N_striing (const char T[])
470.N_striing (const char &Gover)
471.N_striing (const N_striing &C)
472.N_striing & operator= (const N_striing &C)
473.bool operator== (const N_striing &C)
474.bool operator!= (const N_striing &C)
475.bool operator> (N_striing &C)
476.bool operator< (N_striing &C)
477.N_striing operator+ (const N_striing &c)
478.N_striing operator+ (const char &c)
479.N_striing operator+ (const char c[])
480.N_striing operator+ (const int &i)
481.N_striing & operator+= (const N_striing &c)
482.N_striing & operator+= (const char &c)
483.N_striing & operator+= (const char c[])
484.N_striing & operator+= (const int &i)
485.char operator[] (int values)
486.const char * m_c_str ()
```

487.N_striing & m_itoa (long long i)
 488.long long m_atoi (int variable_start, int variable_stop)
 489.N_striing & m_push_back (const char &Gover)
 490.N_striing & m_push_back (const char Gover[])
 491.N_striing & m_push_front (const char &Gover)
 492.N_striing & m_push_front (const char Gover[])
 493.N_striing & m_insert (int value, const char Gover)
 494.N_striing & m_insert (int value, const char Gover[])
 495.N_striing & m_swap (const char &Gover_old, const char &Gover_new)
 496.N_striing & m_swap (const char Gover_old[], const char Gover_new[])
 497.N_striing & m_pop_back ()
 498.N_striing & m_pop_front ()
 499.N_striing m_clear ()
 500.N_striing & m_erase (int i)
 501.N_striing & m_erase_ray (int value_front, int value_back)
 502.N_striing & m_erase_ray (int value_front)
 503.N_striing m_cut (int value_front, int value_back)
 504.bool m_wchat_char (const char &variable)
 505.bool m_wchat_char (const char variable[])
 506.N_striing m_cut (int value_front)
 507.N_striing & m_shift (int i, const char &value)
 508.int m_size ()
 509.N_striing & m_getline (std::ifstream &is)
 510.N_striing (const char T[])
 511.N_striing (const char &Gover)
 512.N_striing (const N_striing &C)
 513.N_striing & operator= (const N_striing &C)
 514.bool operator== (const N_striing &C)
 515.bool operator!= (const N_striing &C)
 516.N_striing operator+ (const N_striing &c)
 517.N_striing operator+ (const char &c)
 518.N_striing operator+ (const char c[])
 519.N_striing operator+ (const int &i)
 520.N_striing & operator+= (const N_striing &c)
 521.N_striing & operator+= (const char &c)
 522.N_striing & operator+= (const char c[])
 523.N_striing & operator+= (const int &i)
 524.char operator[] (int values)
 525.const char * m_c_str ()
 526.N_striing & m_itoa (long long i)
 527.long long m_atoi (int variable_start, int variable_stop)
 528.N_striing & m_push_back (const char &Gover)
 529.N_striing & m_push_back (const char Gover[])
 530.N_striing & m_push_front (const char &Gover)
 531.N_striing & m_push_front (const char Gover[])
 532.N_striing & m_insert (int value, const char Gover)
 533.N_striing & m_insert (int value, const char Gover[])
 534.N_striing & m_swap (const char &Gover_old, const char &Gover_new)
 535.N_striing & m_swap (const char Gover_old[], const char Gover_new[])
 536.N_striing & m_pop_back ()
 537.N_striing & m_pop_front ()
 538.N_striing & m_clear ()
 539.N_striing & m_erase (int i)

540.N_striing & m_erase_ray (int value_front, int value_back)
541.N_striing & m_erase_ray (int value_front)
542.N_striing m_cut (int value_front, int value_back)
543.bool m_wchat_char (const char &variable)
544.bool m_wchat_char (const char variable[])
545.N_striing m_cut (int value_front)
546.N_striing & m_shift (int i, const char &value)
547.int m_size ()
548.N_striing & m_getline (std::ifstream &is)

Przyjaciele

549.std::ostream & operator<< (std::ostream &is, N_striing &C)
550.N_striing & operator>> (std::fstream &is, N_striing &C)
551.std::ostream & operator<< (std::ostream &is, N_striing &C)
552.N_striing & operator>> (std::fstream &is, N_striing &C)

Dokumentacja dla tej klasy została wygenerowana z plików:

553.Data/narzedzia/striing.h
554.Data/narzedzia/Striing.cpp

Dokumentacja szablonu klasy N_vektor< T >

Metody publiczne

555.N_vektor (int i)
556.N_vektor (const N_vektor &V)
557.N_vektor & operator= (const N_vektor &V)
558.bool operator== (const N_vektor &V)
559.bool operator!= (const N_vektor &V)
560.T operator[] (int values)
561.N_vektor & m_push_back (T inside)
562.N_vektor & m_push_front (T inside)
563.N_vektor & m_pop_back ()
564.N_vektor & m_pop_front ()
565.N_vektor & m_insert (int value, T inside)
566.N_vektor & m_erase (int value)
567.N_vektor m_close ()
568.int m_size ()

Dokumentacja dla tej klasy została wygenerowana z pliku:

569.Data/narzedzia/Vektor.h

570. Indeks
INDEX

6. RAPORT Z TESTÓW MODUŁÓW

Testowane przez M. Marchelewicz (v. 1.01):

Program w wersji 1.01 kompiluje się prawidłowo. Wszystkie dołączone headery + biblioteki string oraz vector ze sobą współgrają. Zostało przetestowane wyświetlanie danych na ekran. Dane wyświetlane są prawidłowo, co oznacza że konstruktory również są zaimplementowane prawidłowo.

Poniżej kod z pierwszej fazy testowania:

```
#include <iostream>
#include "Data\Personal\id.h"
// #include "Data\Personal\first_name.h"
// #include "Data\Personal\last_name.h"
// #include "Data\Personal\gender.h"
int main()
{
    //N_striing N = "Alek";
    //N_striing N = "Nowak";
    N_striing N = "36";
    //C_gender F,G(true);
    //C_first_name F(N);
    //C_last_name F(N);
    C_id F(N);

    F.m_get_contens(N);

    std::cout << "ID: " << F.m_set_contens() << "\n";
    //std::cout << "Surname: " << F.m_set_contens() << "\n";
    //std::cout << "Name: " << F.m_set_contens() << "\n";
    //std::cout << "gender:\t" << F.m_set_contens() << "\t" << G.m_set_contens() << "\n";
    return 0;
}
```

Testowane przez Ł. Witek vel Witkowski (v. 1.1):

Program w wersji 1.1 kompiluje się prawidłowo. Wszystkie dołączone headery + biblioteki string oraz vector ze sobą współgrają. Dodatkowo zaimplementowano nowe headery (tree.h). Zostało przetestowane wyświetlanie danych na ekran. Dane wyświetlane są prawidłowo, co oznacza że konstruktory również są zaimplementowane prawidłowo.

```

1  #include <iostream>
2  #include "Data\Databases\tree.h"
3  int main()
4  {
5      C_id ID(22);
6      C_human *Human = new C_human;
7      Human->m_shift_id(ID);
8      Human->m_get_gender(true);
9      N_string F("Lukasz"), L("Witek");
10     Human->m_get_first_name(F);
11     Human->m_get_last_name(L);
12     Human->m_get_last_name(F);
13     C_human H1(*Human);
14     C_tree T(H1);
15     H1.m_get_gender(false);
16     C_human H2(T.m_set_human_index());
17     delete Human;
18     std::cout << "Human: " << H1.m_set_id().m_set_contens() << "\t" << H1.m_set_first_name().m_set_contens() <<
19     "\t" << H1.m_set_gender().m_set_contens() << "\t" << H1.m_set_last_name().m_set_contens() <<
20     "\t" << H1.m_set_last_name(1).m_set_contens() << "\n";
21     std::cout << "Human: " << H2.m_set_id().m_set_contens() << "\t" << H2.m_set_first_name().m_set_contens() <<
22     "\t" << H2.m_set_gender().m_set_contens() << "\t" << H2.m_set_last_name().m_set_contens() <<
23     "\t" << H2.m_set_last_name(1).m_set_contens() << "\n";
24
25     return 0;
26 }

```

Kolejna próba kompilacji + testowanie pól z datami (dzień, miesiąc, rok). Aplikacja zwraca prawidłowe wartości liczbowe.

```

1  #include <iostream>
2  #include "Data\Databases\tree.h"
3  int main()
4  {
5      C_day day;
6      C_month month;
7      C_year year;
8      N_string data;
9      int X, Return = 0;
10     data = day.m_what_type();
11     for (X = 0; X < data.m_size(); X++)
12         Return += (int)data[X];
13     int Return1 = 0;
14     data = month.m_what_type();
15     for (X = 0; X < data.m_size(); X++)
16         Return1 += (int)data[X];
17     int Return2 = 0;
18     data = year.m_what_type();
19     for (X = 0; X < data.m_size(); X++)
20         Return2 += (int)data[X];
21     std::cout << "day: " << Return << "\nmonth: " << Return1 << "\nyear" << Return2 << "\n";
22     return 0;
23 }

```

I kolejna kompilacja, po dołączeniu headera **tree.h**. Pola z danymi osobowymi współgrają z w/w biblioteką.

```

1  #include <iostream>
2  #include "Data\Databases\tree.h"
3  int main()
4  {
5      C_id Id; C_first_name First; C_last_name Last; C_gender gender; N_string data;
6      int X, Return3 = 0, Return4 = 0, Return5 = 0, Return6 = 0;
7      data = Id.m_what_type();
8      for (X = 0; X < data.m_size(); X++)
9          Return3 += (int)data[X];
10     data = First.m_what_type();
11     for (X = 0; X < data.m_size(); X++)
12         Return4 += (int)data[X];
13     data = Last.m_what_type();
14     for (X = 0; X < data.m_size(); X++)
15         Return5 += (int)data[X];
16     data = gender.m_what_type();
17     for (X = 0; X < data.m_size(); X++)
18         Return6 += (int)data[X];
19     std::cout << "\nId: " << Return3 << "\nFirst_name: " << Return4 <<
20     "\nLast_name: " << Return5 << "\nGender: " << Return6 << "\n";
21     return 0;
22 }

```

Testowane przez M. Marchelewicz, Ł. Witek (v. 1.1):

Testowane zostały moduły **application.h** oraz **application_txt.h**. Oba moduły współpracują ze sobą. Program się kompiluje. Tworzy się okno o ustalonej wartości, czcionka automatycznie się powiększa.

```
Drzewo_genealogiczne (Global Scope) main()
73 C_last_name L1, L2;
74 data = "acb";
75 L1.m_get_contens(data);
76 data = "abc";
77 L2.m_get_contens(data);
78 if (L1 > L2) std::cout << "dobrze\n"; else std::cout << "zle\n"; /*
79
80 C_aplication_txt AP; // test menu w aplikacji - działa!!!
81 AP.SetWindow(100, 45);
82 AP.CreateLogo();
83 AP.MainMenu();
84
85
86 //test na dzialanie C_date
87 /*C_date date13('/');
88 date13.m_shift_day(12);
89 date13.m_shift_month(10);
90 date13.m_shift_year(1991);
91 std::cout << date13.m_set_DD_MM_YYYY()<<'\\n';
92
93 //test na poskie znaki
94 C_first_name test101;
95 N_string fff = "Łukasz";
96 test101.m_get_contens(fff);
97 std::cout << "test 101:"<< test101<<"\\n\\n";
98
99 */
```

Moduł **date.h** działa prawidłowo. Data wyświetla się poprawnie. Znak „/” oddziela prawidłowo dzień, miesiąc i rok. Polskie znaki również już działają. Pomogła poprawa pętli for.

```
79
80 /*C_aplication_txt AP; // test menu w aplikacji - działa!!!
81 AP.SetWindow(100, 45);
82 AP.CreateLogo();
83 AP.MainMenu();
84 */
85
86 //test na dzialanie C_date
87 C_date date13('/');
88 date13.m_shift_day(12);
89 date13.m_shift_month(10);
90 date13.m_shift_year(1991);
91 std::cout << date13.m_set_DD_MM_YYYY()<<'\\n';
92
93 //test na poskie znaki
94 C_first_name test101;
95 N_string fff = "Łukasz";
96 test101.m_get_contens(fff);
97 std::cout << "test 101:"<< test101<<"\\n\\n";
98
99
```

Kolejne metody w module **date.h** również wyświetlają poprawnie daty (dla zmiennych typu int). Wszystkie cztery metody działają prawidłowo.

```
83 | AP.MainMenu();  
84 | */  
85 |  
86 | //test na dzialanie C_date  
87 | C_date date13('/');  
88 | date13.m_shift_day(12);  
89 | date13.m_shift_month(10);  
90 | date13.m_shift_year(1991);  
91 | //std::cout << date13.m_set_DD_MM_YYYY()<<'\n';  
92 | //std::cout << date13.m_set_MM_DD_YYYY() << "\n";  
93 | //std::cout << date13.m_set_YYYY_MM_DD() << "\n";  
94 | std::cout << date13.m_set_YYYY_DD_MM() << "\n";  
95 |
```

Daty również poprawnie wyświetlają się, gdy wprowadzane są za pomocą własnej biblioteki string (dla zmiennych typu string).

```
85 |  
86 | //test na dzialanie C_date  
87 | C_date date13('/');  
88 | //date13.m_shift_day(12);  
89 | //date13.m_shift_month(10);  
90 | //date13.m_shift_year(1991);  
91 | date13.m_shift_day("12");  
92 | date13.m_shift_month("10");  
93 | date13.m_shift_year("1991");  
94 | //std::cout << date13.m_set_DD_MM_YYYY()<<'\n';  
95 | //std::cout << date13.m_set_MM_DD_YYYY() << "\n";  
96 | //std::cout << date13.m_set_YYYY_MM_DD() << "\n";  
97 | std::cout << date13.m_set_YYYY_DD_MM() << "\n";  
98 |
```

Równocześnie można stwierdzić, że działają moduły **day.h**, **month.h**, **year.h**, gdyż ściśle współpracują one z modulem **date.h**. Poniżej test kluczowych metod w/w modułów.

```
105 |  
106 | C_day day2;  
107 | N_string ddd = "20";  
108 |  
109 |  
110 | day2.m_get_day(ddd);  
111 | std::cout << ddd << "\n";  
112 |  
113 | C_month month2;  
114 | N_string mmm = "10";  
115 |  
116 |  
117 | month2.m_get_month(mmm);  
118 | std::cout << mmm << "\n";  
119 |  
120 | C_year year2;  
121 | N_string yyy = "1999";  
122 |  
123 |  
124 | year2.m_get_year(yyy);  
125 | std::cout << yyy << "\n";  
126 |
```

Kolejny test z modułów **first_name.h**, **last_name.h**, **gender.h**. Wszystkie pola prawidłowo się wyświetlają. Dodatkowo działają polskie litery. W **gender.h** nie działa jeden warunek. Będzie wymagał poprawki.

```
100 //test na poskie znaki
101 C_first_name test101;
102 N_striing fff = "Łukasz";
103 test101.m_get_contens(fff);
104 std::cout << "test 101:" << test101 << "\n";
105
106 C_last_name test102;
107 N_striing fff2 = "Mikuła";
108 test102.m_get_contens(fff2);
109 std::cout << "test 102:" << test102 << "\n";
110
111 C_gender test103;
112 //N_striing fff3 = "Men";
113 //N_striing fff3 = "men";
114 //N_striing fff3 = "1";
115 //N_striing fff3 = "true";
116 N_striing fff3 = "Women";
117 //N_striing fff3 = "women";
118 //N_striing fff3 = "0";
119 //N_striing fff3 = "False"; --> nie działa!!!
120 test103.m_get_contens(fff3);
121 std::cout << "test 103:" << test103 << "\n";
122
```

Moduł **id.h** działa poprawnie. Wyświetla nr rekordu, nawet bardzo dużą liczbę (na jaką pozwala zakres typu).

```
123
124 C_id test104;
125 //test104.m_get_contens(45);
126 //test104.m_get_contens(4543434);
127 //test104.m_get_contens(05);
128 std::cout << test104.m_set_contens();
129
```

Podstawowe metody w module **children.h** działają poprawnie. Współpracują one z modulem **id.h**.

```
148
149
150 year2.m_get_year(yyy);
151 std::cout << yyy << "\n";
152 */
153
154 C_children chil;
155
156 chil.m_get_complete_content(34, 3);
157
```

Moduł **relation.h** również działa prawidłowo. Zawiera on metody czysto wirtualne, więc bezpośrednie przetestowanie nie jest w tej chwili możliwe.

```

21 class C_relation
22 {
23 public:
24     C_relation();
25     C_relation(N_string realt);
26     C_relation(const C_relation &relat);
27     C_relation& operator=(const C_relation &relat);
28     bool operator==(const C_relation &relat);
29     bool operator!=(const C_relation &relat);
30     virtual void m_get_id(C_id &id) = 0;
31     virtual C_id m_set_id() = 0;    //~C_relation();
32     virtual ~C_relation();
33     N_string m_what_type();
34     virtual int m_set_variable() = 0;
35     virtual void m_get_complete_content(N_string data) = 0;
36     virtual void m_get_complete_content(C_id index, C_id value) = 0;

```

Testowane przez M. Marchelewicz, Ł. Witek (v. 1.2):

Test ładowania nazwiska z pliku przeszedł pomyślnie.

The image shows a Visual Studio IDE with a C++ project named 'Drzewo_genealogiczne'. The code in 'engineer.cpp' defines a structure for a person and a function to build a tree. The output window shows the execution results, including the names of the people in the tree and the final structure of the tree.

Code Snippet (engineer.cpp):

```

119     data.m_push_back(s_government_personality[i]);
120     if (data == n_last_name)
121     {
122         str.m_clear();
123         for (j = i+1; j < s_government_personality.size(); j++)
124         {
125             if (!s_government_personality[j].m_is_alive)
126             {
127                 str.m_push_back(s_government_personality[j].m_name);
128                 continue;
129             }
130             Last.m_get_contens(str);
131             std::cout << Last << "\n";
132             Vektor.m_push_back(Last);
133             i = j;
134             break;
135         }
136     }
137     return Vektor;
138 }
139 //przetestowac czy dziala
140 C_gender C_government_personality::m_set_value_gender(C_gender Gender);
141

```

Output Window:

```

99 %
C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_Semestr_IV\Projekt_4\Drzewo_genealogiczne\Drzewo_genealogiczne.vcxproj -> C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_Semestr_IV\Projekt_4\Drzewo_genealogiczne\Drzewo_genealogiczne.vcxproj (x64)
Build
Show output from: Build
1> C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_Semestr_IV\Projekt_4\Drzewo_genealogiczne\Drzewo_genealogiczne.vcxproj -> C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_Semestr_IV\Projekt_4\Drzewo_genealogiczne\Drzewo_genealogiczne.vcxproj (x64)
1> Drzewo_genealogiczne.vcxproj -> C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_Semestr_IV\Projekt_4\Drzewo_genealogiczne\Drzewo_genealogiczne.vcxproj (x64)
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
<d143$0Lukasz&0Witek&0Witkowski!0Men>
<d143$0Lukasz&0Witek&0Witkowski!0Men>
<d144$0Lukasz&0Witek&0Witkowski!0Men>
<d145$0Lukasz&0Witek&0Witkowski!0Men>
<d146$0Lukasz&0Witek&0Witkowski!0Men>
<d147$0Lukasz&0Witek&0Witkowski!0Men>

```

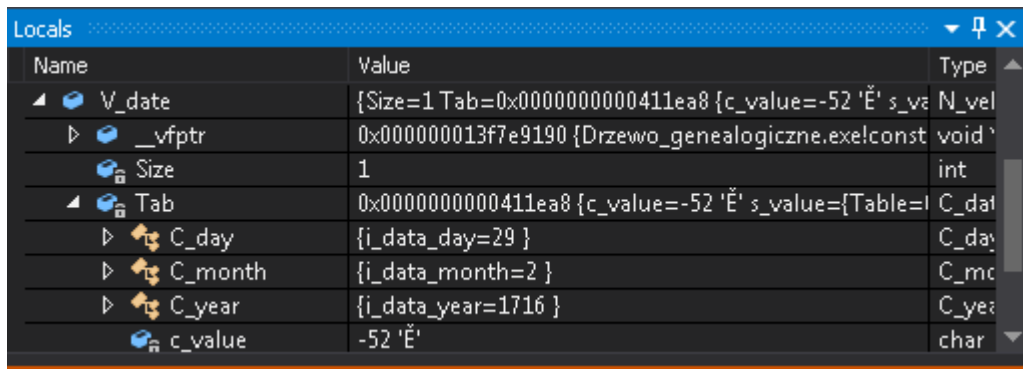
[illegible]

```

date13.m_get_type(data);
human.m_get_first_name(first);
human.m_get_last_name(last);
human.m_get_last_name(last1);
human.m_get_gender(gender);
human.m_get_date(date15[0]);
C_element element(human);
C_children children(human.m_set_id());
C_id id(10055);
C_grandchildren gchildren(human.m_set_id());
C_grandparents gparents(human.m_set_id());
C_parent parent(human.m_set_id());
C_partner partner(human.m_set_id());
C_sibling sibling(human.m_set_id());
children.m_get_id(id);
gchildren.m_get_id(id);
gparents.m_get_id(id);
parent.m_get_id(id);
partner.m_get_id(id);
sibling.m_get_id(id);
element.m_get_children(children);
element.m_get_grandchildren(gchildren);
element.m_get_grandparents(gparents);
element.m_get_parent(parent);
element.m_get_partner(partner);
element.m_get_sibling(sibling);
Engin.m_new_element(element, false);
C_element EEE(Engin.m_create_element(0));
C_human HHH(Engin.m_create_human(0));
if (human == HHH) std::cout << "\nPrawidlowa stworzony human:\n";
else std::cout << "\nblad z humanem:\n";
if (element == EEE) std::cout << "\nPrawidlowa stworzony element:\n";
else std::cout << "\nblad z elementem:\n";

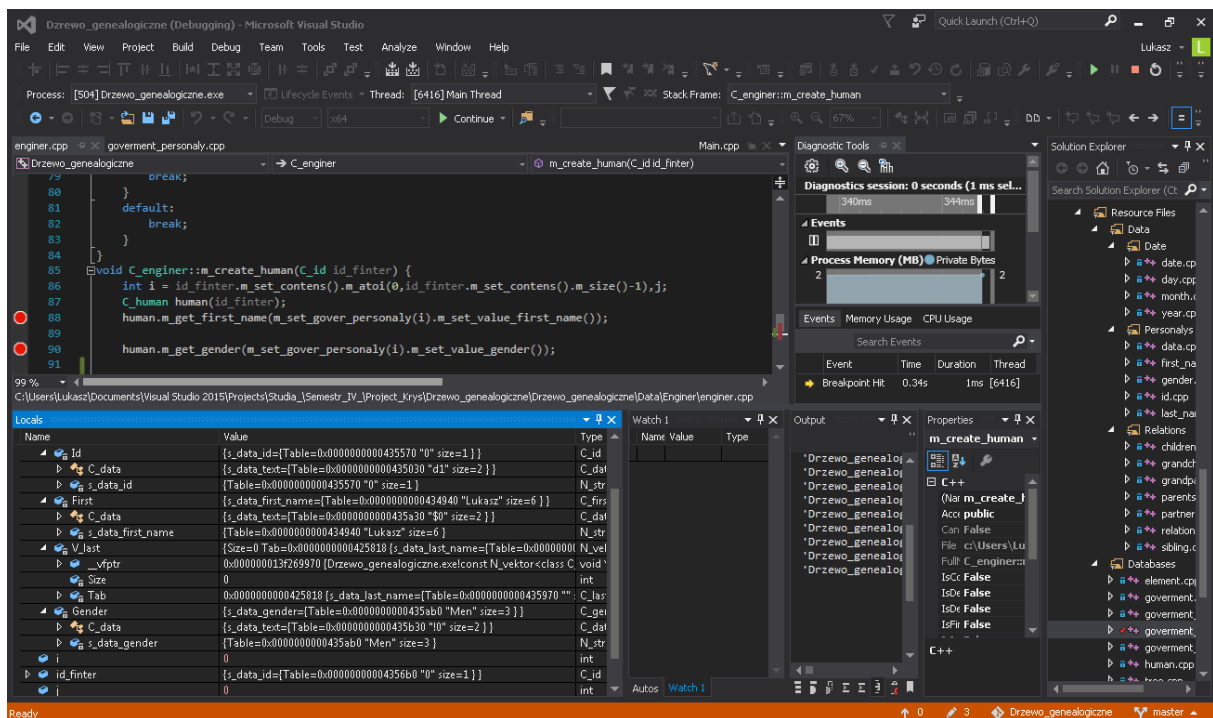
```

Test na wczytanie dat (dnia, miesiąca i roku). Wszystko przebiegło poprawnie w debuggerze.



Name	Value	Type
V_date	{Size=1 Tab=0x000000000411ea8 {c_value=-52 'Ě' s_value=N_vel	N_vel
__vfptr	0x000000013f7e9190 {Drzewo_genealogiczne.exe!const	void *
Size	1	int
Tab	0x00000000000411ea8 {c_value=-52 'Ě' s_value={Table=	C_da
C_day	{i_data_day=29 }	C_da
C_month	{i_data_month=2 }	C_mo
C_year	{i_data_year=1716 }	C_ye
c_value	-52 'Ě'	char

Testowanie na poprawne ładowanie danych do obiektu human.



The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays the `engineer.cpp` file, specifically the `m_create_human` function. The function is currently paused at line 80, which is a `break;` statement. The code includes comments and function calls like `human.m_get_first_name` and `human.m_get_gender`.
- Locals Window:** Shows the state of local variables. Key variables include `Id` (a `C_id` object), `C_data` (a `C_da` object), `First` (a `C_firs` object), `C_data` (a `C_da` object), `s_data_first_name` (a `N_str` object), `s_data_last_name` (a `N_str` object), `__vfptr` (a `void *`), `Size` (an `int`), `Tab` (a `C_las` object), `Gender` (a `C_ge` object), `C_data` (a `C_da` object), `s_data_gender` (a `N_str` object), `i` (an `int`), `id_finter` (a `C_id` object), and `i` (an `int`).
- Diagnostic Tools:** Shows the `Events` window with a `Breakpoint Hit` event at line 80, column 1, with a duration of 0.34s and 1ms.
- Solution Explorer:** Displays the project structure, including files like `data.cpp`, `day.cpp`, `month.cpp`, `year.cpp`, `personals`, `relations`, and `databases`.
- Output Window:** Shows the output of the `m_create_human` function, including the `Accor public` and `Can False` attributes.

7. RAPORT Z TESTÓW MODUŁÓW

DATA	TESTER	PRZEDMIOT TESTU	UWAGI
04.2017	ŁW, MM	Testowanie własnej biblioteki string	działa prawidłowo
04.2017	ŁW	Testowanie własnej biblioteki vector	działa prawidłowo
17.05.2017	MM	Testowanie menu	nie jest to wersja finalna
18.05.2017	MM, ŁW	Test modułów z folderu Date	brak
18.05.2017	MM, ŁW	Test modułów z folderu Personalys	potrzebne kilka poprawek
18.05.2017	MM, ŁW	Test modułów z folderu Interface	potrzebne kilka poprawek
05.2017	ŁW, MM	Test modułów z folderu Relations	
05.2017	ŁW	Test modułów z folderu Databases	
05.2017	ŁW	Test modułów z folderu Enginer	
25.05.2017	ŁW	Test wczytywania z pliku	
25.05.2017	ŁW	Test zapisu do pliku	
		Testowanie menu	
		Test wyświetlania drzewa	
		Test wyszukiwania osoby	
		Test działania relacji	
		Test menu + wszystkich podopcji	
		Testy końcowe	

8. DOKUMENTACJA ADMINISTRATORA

Instalacja:

Po ściągnięciu programu ze strony, należy go rozpakować do wybranego przez użytkownika folderu. Domyślnie program rozpakowany zostanie do katalogu **C:/tree**. Lokalizacja programu nie ma wpływu na poprawność jego działania. Program po rozpakowaniu jest od razu gotowy do pracy. Program jest uruchamiany za pośrednictwem pliku **genealogy_inop.exe**.

Kompletny schemat dostępnych funkcji programu,

Obsługa programu – klawisze:

góra/dół	-	poruszanie się po menu
Enter	-	zatwierdzenie wyboru
Spacja	-	powrót do menu

Menu główne i podmenu: Create Tree, Load Tree

- 1. Create Tree
 - Create New Tree
 - Import Tree
- 2. Load Tree
 - Display Tree
 - Edit Tree
 - Export Tree
 - Exit
- 3. Exit

Menu Display Tree

- 1. Display from the oldest
- 2. Search
 - Search by personal data
 - Search by date
 - Exit
- 3. Exit

Menu edycji drzewa

<Nazwa drzewa>

1. Add a person
 - podaj imię, nazwisko, daty: narodziny, śmierć
 - dodaj datę ślubu (podmenu → dodaj osobę) i datę rozwodu
 - relacja inna: data (podmenu → dodaj osobę, → dodaj dzieci)
 - dodaj do (→ podaj imię i nazwisko) jako 'potomek'/'przodek'
2. Find a person
 - podaj imię i nazwisko
3. Add a relation.
 -
 -
4. Edit a relation.
 -
 -
5. Save a tree.
 - podaj nazwę (domyślnie podstawia się nazwa wczytanego drzewa)
6. Exit.

Zasady wypełniania pól i opis błędów.

Format danych:

Imiona i nazwiska należy wprowadzać bez polskich znaków diakrytycznych. Daty należy podać w formacie ddmrrrr.

Wprowadzanie dat:

Komunikat błędu:

'Sprawdź datę – rok RRRR nie był przestępny'.

Uwarunkowania historyczne skutkujące zmianą w sposobie liczenia czasu, (m. in. brak dat z zakresu od 05.10.1582 do 14.10.1582r. dla kalendarza gregoriańskiego) nie są uwzględnione.

Obecna wersja programu dopuszcza wprowadzanie dat od roku 01.01.0001. Lata przestępne liczone są wg schematu: suma liczb w roku podzielna przez '4' . Nie został uwzględniony warunek uzupełniający: lata podzielne przez 100 i niepodzielne przez 400 również będą mogły posiadać luty mający 29 dni. Pozwala to na kompatybilność z dwoma najpopularniejszymi na świecie systemami kalendarzowymi – kalendarzem Juliańskim i Gregoriańskim.

Śluby

Komunikat: 'Przekroczenie minimalnego wieku'

Minimalny wiek dla zawarcia małżeństwa w programie ustalony został na 12 lat dla kobiety i 13 lat dla mężczyzny (zgodnie z najczęstszymi uwarunkowaniami historycznymi).

Różnica wieku między małżonkami

Komunikat: 'Przekroczenie maksymalnego wieku'.

Maksymalna różnica wynosi 100 lat (przyjęta ogólnie przez autorów programu).

Dodawanie dzieci

Dzieci można dodawać niezależnie od daty ślubu, ponieważ data ślubu nie zawsze jest wcześniejsza od daty narodzin dziecka.

Dodawanie kolejnego dziecka

Komunikat: 'Błędna data narodzin kolejnego dziecka'.

Minimalny okres między dodaniem 'kolejnego potomka dla tej samej matki ustalany jest na minimum 175 dni (25 tygodni) (przyjęte przez autorów jako wiek z minimalną szansą na przeżycie dla wcześniaków).

W przypadku ciąży wielokrotnej dzieci można dodać tego samego dnia lub, jeśli poród był w nocy, z różnicą maksymalnie jednego dnia.

Dodawanie kolejnego dziecka – zdublowane imiona

Komunikat: 'Dziecko o imieniu *imię* i nazwisku *nazwisko* już istnieje '.

Nie można dodać jednocześnie dwóch tych samych osób (dzieci tych samych rodziców) z tą samą datą lub różną datą urodzenia, ale żyjących jednocześnie i mających to samo imię i nazwisko.

Gdy jedno dziecko zmarło i po jego śmierci urodziło się kolejne, wtedy możliwe będzie powtórne wprowadzenie tego samego imienia i nazwiska.

Maksymalny wiek matki

Komunikat: 'Przekroczony maksymalny wiek dla urodzenia dziecka'.

Na moment pisania programu maksymalny znany wiek przyjęty przez autorów, dla matki rodzącej dziecko, wynosi 67 lat.

Dodawanie nowej osoby

Standardowo należy podać imię, nazwisko, datę urodzenia i śmierci. W przypadku danych niepełnych, w miejsce imienia i nazwiska można wpisać „nieznany” lub „nieznana”, daty natomiast należy podać w przybliżeniu, aby umiejscowić przodka w określonym pokoleniu.

Dzieci nieślubne

W przypadku nieznanego rodzica lub dziecka, które nie miało rodziców będących w związku małżeńskim, należy użyć opcji „relacja inna”, oraz dodanie w ten sposób matki/ojca (znanego lub nie). Dane drugiego rodzica są nieznane, w pola imię, nazwisko można wpisać nieznany/nieznana; daty należy podać w przybliżeniu, aby umiejscowić taką osobę w drzewie.

.....

.....

.....

9. DOKUMENTACJA UŻYTKOWNIKA

(Ł. Janus)

9.1. INFORMACJE WSTĘPNE

Program pozwala na stworzenie prostego drzewa genealogicznego, wraz z konsolową wizualizacją trzech pokoleń.

W ramach programu będzie możliwe:

- dodawanie dowolnej ilości osób do drzewa
- edycja danych już dodanej osoby
- usuwanie błędnie dodanej osoby

9.2. PORUSZANIE SIĘ PO MENU

Do poruszania się po menu programu użyj strzałek „w górę” i „w dół”. Aby wejść w podmenu lub zatwierdzić wpisane dane, użyj klawisza „Enter”. W celu powrotu do menu głównego, wykorzystaj klawisz „spacja”.

9.3. URUCHOMIENIE PROGRAMU

W celu uruchomienia programu, kliknij dwukrotnie ikonę genealogi_inop.exe, która znajduje się na pulpicie lub w folderze c:/tree

Menu Główne

Menu główne pojawia się po uruchomieniu programu. Oto jego elementy:

1. Create Tree – umożliwia stworzenie nowego drzewa.
2. Load Tree – umożliwia wczytanie wcześniej zapisanego drzewa
3. Exit – wyjście z programu.

Create Tree

Po wybraniu tej opcji program poprosi o wpisanie nazwy dla nowego drzewa (Create New Tree) lub wczytanie z pliku innego drzewa (Import Tree). Następnie program przechodzi do menu edycji drzewa, umożliwiając zarządzanie osobami i relacjami:

- Add a person – za pomocą tej opcji można dodać osobę do drzewa (pierwszą lub kolejną)
- Find a person – umożliwia wyszukanie osoby w drzewie
- Add a relation – opcja pozwalająca na dodanie nowej relacji (osoby)
- Edit a relation – umożliwia edycję istniejącej relacji
- Exit – wyjście z programu

Menu Load Tree

W tym miejscu można wyszukać osobę i wczytać drzewo wg osób, które zostały w nim umieszczone, jak również przejście do edycji istniejącego drzewa i export drzewa do pliku (np. w celu przeniesienia go na inny komputer).

1. Display Tree – pozwala wyświetlić drzewo z zapisanych wcześniej projektów
2. Edit Tree – edycja drzewa, po wybraniu program przechodzi do menu edycji drzewa analogicznie, jak po opcji Create Tree.
3. Export Tree – umożliwia usunięcie istniejącego drzewa.
4. Exit – wyjście z programu

Menu Display Tree

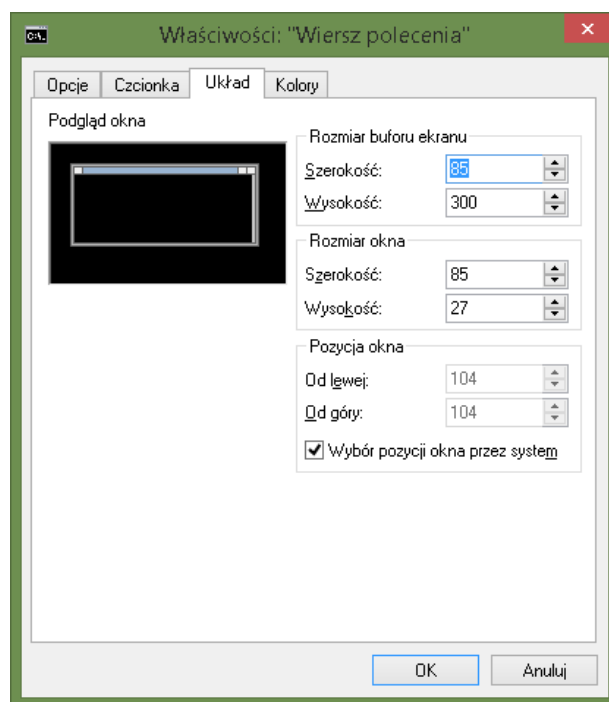
1. Display from the oldest – wyświetlanie drzewa od najstarszego przodka
2. Search – wyszukiwanie osób po danych personalnych lub na podstawie daty (np. data urodzenia, data ślubu)
 - a) Search by personal data
 - b) Search by date
 - c) Exit
3. Exit.

9.4. PORADY OGÓLNE

Wyświetlanie drzewa

Program pracuje w konsoli systemu Windows, która ma ograniczenia dotyczące rozdzielczości. Program automatycznie ustawia wielkość okna i czcionkę, lecz gdyby całe drzewo nie mieściło się wszerek na ekranie, kliknij prawym przyciskiem w lewy górny róg konsoli. Powinno pojawić się menu kontekstowe. Wybierz opcję „Właściwości” i następnie przejdź na zakładkę „Układ”.

W zakładce „Układ” ręcznie zmień „Szerokość okna” do zadowalającej Cię wielkości. Jeśli ta opcja nie przynosi efektu, trzeba w tej samej zakładce zwiększyć najpierw „Szerokość” pod pozycją „Rozmiar buforu ekranu”.



LEGENDA

Informacje dostępne do wprowadzenia/edycji na każdym członku drzewa (wraz z oznaczeniami):

- **narodziny** (%)
- **ślub** (*)
- **zgon** (+)

Przykładowy zapis:

Jan Kowalski % 01.01.1900 + 02.02.1930	Związki: * 1919, Janina Nowak * 1921, Jadwiga Nowakowska
	Relacje inne: 1923 Anna Kowalewska (potomkowie)

9.5. SCHEMAT DOSTĘPNYCH OPCJI

Menu główne i podmenu: Create tree, Load Tree

- 1. Create Tree
 - Create New Tree
 - Import Tree
- 2. Load Tree
 - Display Tree
 - Edit Tree
 - Export Tree
 - Exit
- 3. Exit

Menu Display Tree

- 1. Display from the oldest
- 2. Search
 - Search by personal data
 - Search by date
 - Exit
- 3. Exit

Menu edycji drzewa

<Nazwa drzewa>

1. Add a person
 - podaj imię, nazwisko, daty: narodziny, śmierć
 - dodaj datę ślubu (podmenu → dodaj osobę) i datę rozwodu
 - relacja inna: data (podmenu → dodaj osobę, → dodaj dzieci)
 - dodaj do (→ podaj imię i nazwisko) jako 'potomek'/'przodek'
2. Find a person
 - podaj imię i nazwisko
3. Add a relation.
 -
 -
4. Edit a relation.
 -
 -
5. Save a tree.
 - podaj nazwę (domyślnie podstawia się nazwa wczytanego drzewa)
6. Exit.

10. DOKUMENTACJA Z FAZY KONSERWACJI

Podstawowy rezultat.

Poprawiony kod, projekt, model i specyfikacja wymagań.

11. RAPORT Z POSTĘPÓW PRODUKCJI OPROGRAMOWANIA (PROGRESS REPORT)

11.1. Tablica postępów

Document: **FamilyTree_Documentation**

FFFFFF Program tworzący drzewo genealogiczne

Originator: M. Marchelewicz Recipient: A. Kryś

Version history:

Data	Stworzył:	Opis
2017/05/04	M. Marchelewicz	Original
..		

Ref.as per FFFFFF	Build	Description (Opis)	Clarified (Wyjaśnione)	Design I	Accept	Design II	Implement	QA	Test	Remarks (Uwagi)
Platform										
1.1.1	[B1]	Hardware	Y	Y	Y	Y	NA	Y		
1.1.2	[B1]	OS	Y	Y	Y	Y	NA	Y		
1.1.3	[B1]	Compiler	Y	Y	Y	Y	NA	Y		
Displayed objects										
2.1.	[B1]	Adaptation data (see Section 3),	Y	Y	N	Y	Y	Y		
2.2.	[B2]	Display name, surname	Y	Y	Y	Y	Y	Y		

Ref.as per FFFFF	Build	Description (Opis)	Clarified (Wyjaśnione)	Design I	Accept	Design II	Implement	QA	Test	Remarks (Uwagi)
2.2.1	[B2]	Display ID	Y	Y	Y	Y	Y	Y		
2.2.2.	[B2]	Display gender	Y	Y	Y	Y	Y	Y		
2.2.3.	[B2]	Display menu	Y	Y	Y	Y	Y	Y		
2.2.4.	[B3]	Reading from files	Y	Y	N	N	N			
2.2.5.	[B4]	Saving data to files	Y	Y	N	N	N			
Configuration file										
3.1.	[B1]	Using own string library	Y	Y	Y	Y	Y			
3.2.	[B2]	Using own vector library	Y	Y	Y	Y	Y			
3.3.1.	[B3]	Application name	Y	Y	Y	Y	Y			
3.3.2.	[B3]	Application menu	Y	Y	N	Y	Y			
3.3.3.3.	[B3]	Color menu	Y	Y	Y	Y	Y	Y		
3.3.3.4	[B3]	Font Size	Y	Y	Y	Y	Y	Y		
3.3.3.4	[B3]	Window Size	Y	Y	Y	Y	Y	Y		
Adaptation data										

Ref.as per FFFFFF	Build	Description (Opis)	Clarified (Wyjaśnione)	Design I	Accept	Design II	Implement	QA	Test	Remarks (Uwagi)
4.1.1.	[B1]	Personal Data	Y	Y	Y	Y	Y			
4.1.2.	[B2]	Relations	Y	Y	N	N	N			
4.1.3.	[B1]	Dates	Y	Y	Y	Y	Y			

12. DOKUMENTACJA QA (kontrola jakości)

12.1. Tabele QA

Lista testowanych klas:

<ul style="list-style-type: none">• N_string• N_vektor• C_data• C_id• C_first_name• C_last_name• C_gender• C_date• C_day• C_month• C_year• C_children• C_grandchildren• C_grandparents• C_parent• C_partner• C_relation• C_sibling• C_element• C_government• C_government_date• C_government_personalys• C_government_relation• C_human• C_tree• C_engineer• C_save_load• C_sl_date• C_sl_personalys• C_sl_relations• C_aplication• C_aplication_txt	
---	--

12.2. Zgodność kodu ze standardami w formie „QA class document” zawierający każdą klasę i potwierdzający jej zgodność ze standardem

Jednostkowe testy QA (kolejność zgodna z powyższą tabelą):

Class name: N_string		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: N_vektor		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_data		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
	podpis	
	Mateusz Marchelewicz	

Class name: C_id		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
	podpis	
	Mateusz Marchelewicz	

Class name: C_first_name		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
	podpis	
	Mateusz Marchelewicz	

Class name: C_last_name		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
	podpis	
	Mateusz Marchelewicz	

Class name: C_gender		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
	podpis	
	Mateusz Marchelewicz	

Class name: C_date		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_day		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_month		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_year		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_children		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_grandchildren		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_grandparents		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_parent		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_partner		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_relation		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_sibling		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_element		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_goverment		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_goverment_date		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_government_personaly		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Destruktor	N	nieobowiązkowe, jest wirtualny
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_government_relation		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_human		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_tree		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_engineer		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_save_load		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_sl_date		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_sl_personalys		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_sl_relations		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	Y	
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

Class name: C_aplication		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażen goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Zaimplementowane metody	N	nieobowiązkowe
	podpis	
	Mateusz Marchelewicz	

Class name: C_aplication_txt		
Standard (No. / opis)	Zgodny (Y/N)	UWAGI
Konstruktor domyślny	Y	
Konstruktor z parametrami	Y	
Konstruktor kopiujący	Y	
Przeładowany operator	Y	
Brak wyrażen goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

13. DOKUMENTACJA TESTOWANIA

13.1. Tabela proponowanych testów

DATA	TESTER	PRZEDMIOT TESTU	UWAGI
04.2017		Testowanie własnej biblioteki string	
04.2017		Testowanie własnej biblioteki vector	
05.2017		Testowanie menu	
05.2017		Test modułów z folderu Date	
18.05.2017		Test modułów z folderu Personalys	
05.2017		Test modułów z folderu Interface	
05.2017		Test modułów z folderu Relations	
05.2017		Test modułów z folderu Databases	
05.2017		Test modułów z folderu Enginer	
05.2017		Test wczytywania z pliku	
05.2017		Test zapisu do pliku	
05.2017		Testowanie menu	
06.2017		Test wyświetlania drzewa	
06.2017		Test wyszukiwania osoby	
06.2017		Test działania relacji	
06.2017		Test menu + wszystkich podopcji	
06.2017		Testy końcowe	

13.2. Raport błędów

Nagłówek dokumentu

Rodzaj błędu: krytyczny(1), ważny(2), mało ważny(3), kosmetyczny(4).

DATA	WERSJA KODU	TESTER	OPIS BŁĘDU	RODZAJ BŁĘDU	OSOBA KORYGUJĄCA	UWAGI
26.04	1.0	MM	Dodanie biblioteki <cmath>	1	MM	W fazie początkowej, później już niepotrzebna
28.04	1.01	MM	Edycja menu	4	MM	Dodanie podmenu
2.05	1.02	ŁJ	Zamknięcie pliku oraz zwolnienie pamięci tab. dyn.	2	ŁJ	
2.05	1.1	ŁW	Dodanie wirtualnych destruktorów	2	ŁW	Wymagane gdyż użyto wirtualnych metod
8.05	1.2	MM	Błąd dołączonego headera	1	ŁW	Była podana zła lokalizacja headera
18.05	1.2	MM	Błąd w instrukcji if modułu gender.h	3		
20.05	1.2	MM	Błąd przy dołączeniu headera	2	ŁW	Dołączony header nie był potrzebny
25.05	1.2	ŁW	Błąd przy tworzeniu obiektu human	1	ŁW	

14. INNE