

# **DOKUMENTACJA PRZEDSIĘWZIĘCIA PROGRAMISTYCZNEGO**

## **DRZEWO GENEALOGICZNE**

### **Członkowie Grupy „4z”:**

Łukasz Janus - kierownik grupy

Bartosz Bukowski

Mateusz Marchelewicz

Łukasz Witek vel Witkowski

Łódź, 12 czerwca 2017

# SPIS TREŚCI

<b>1. RAPORT Z FAZY STRATEGICZNEJ .....</b>	<b>4</b>
1.1. CEL PRZEDSIĘWZIĘCIA .....	4
1.2. ZAKRES PRZEDSIĘWZIĘCIA .....	4
1.3. OPIS SYSTEMÓW ZEWNĘTRZNYCH Z KTÓRYMI SYSTEM BĘDZIE WSPÓŁPRACOWAĆ .....	4
1.4. OGÓLNY OPIS WYMAGAŃ .....	4
1.5. OGÓLNY MODEL SYSTEMU .....	4
1.6. OPIS PROPONOWANEGO ROZWIĄZANIA .....	4
1.7. WSTĘPNY PROJEKT PROGRAMU .....	5
1.8. PROJEKT MENU, OPCJI EDYCJI DRZEWA ORAZ ZAPISU DANYCH OSOBY W DRZEWIE .....	5
1.9. OSZACOWANIE KOSZTÓW .....	6
1.10. WSTĘPNY HARMONOGRAM PRAC .....	7
1.11. PROPONOWANA WIZUALIZACJA PROGRAMU .....	8
1.12. DIAGRAM GANTTA .....	9
<b>2. RAPORT Z FAZY OKREŚLANIA WYMAGAŃ .....</b>	<b>10</b>
2.1. WPROWADZENIE .....	10
2.2. OPIS PRZEWIDYWANEJ EWOLUCJI SYSTEMU .....	10
2.3. OPIS WYMAGAŃ FUNKCJONALNYCH .....	10
2.4. OPIS WYMAGAŃ NIEFUNKCJONALNYCH .....	11
2.5. OBSŁUGA WYJĄTKÓW .....	12
<b>3. DOKUMENT Z FAZY ANALIZY .....</b>	<b>13</b>
3.1. DIAGRAM PRZEPŁYWU DANYCH .....	15
3.2. DIAGRAM PRZYPADKÓW UŻYCIA .....	16
<b>4. DOKUMENTACJA FAZY PROJEKTOWANIA .....</b>	<b>17</b>
4.1. LISTA KLAS I POWIĄZANIA MIĘDZY KLASAMI .....	17
4.2. DIAGRAM POWIĄZAŃ MIĘDZY KLASAMI .....	19
4.3. NOTACJA UML .....	20
<b>5. DOKUMENTACJA FAZY IMPLEMENTACJI .....</b>	<b>26</b>
5.1. POLICZONE MODUŁY KLOC, SLOC, KDSI .....	17
5.2. POLICZONE KLASY KLOC, SLOC .....	37
5.3. KOD SKŁADAJĄCY SIĘ Z PRZETESTOWANYCH MODUŁÓW .....	54
<b>6. RAPORT Z TESTÓW MODUŁÓW .....</b>	<b>129</b>
<b>7. HARMONOGRAM TESTÓW MODUŁÓW .....</b>	<b>137</b>
<b>8. DOKUMENTACJA ADMINISTRATORA .....</b>	<b>138</b>
<b>9. DOKUMENTACJA UŻYTKOWNIKA .....</b>	<b>147</b>
9.1. INFORMACJE WSTĘPNE .....	147
9.2. PORUSZANIE SIĘ PO MENU .....	147
9.3. URUCHOMIENIE PROGRAMU .....	147
9.4. PORADY OGÓLNE .....	149
9.5. SCHEMAT DOSTĘPNYCH OPCJI .....	150
<b>10. RAPORT Z POSTĘPÓW PRODUKCJI OPROGRAMOWANIA (PROGRESS REPORT) .....</b>	<b>152</b>
10.1. TABLICA POSTĘPÓW .....	152
<b>11. DOKUMENTACJA QA (KONTROLA JAKOŚCI) .....</b>	<b>155</b>
11.1. TABELA QA .....	155
11.2. ZGODNOŚĆ KODU ZE STANDARDAMI W FORMIE „QA CLASS DOCUMENT” .....	156

<b>12.</b>	<b>DOKUMENTACJA TESTOWANIA .....</b>	<b>167</b>
12.1.	TABELA PROPONOWANYCH TESTÓW.....	167
12.2.	RAPORT BŁĘDÓW .....	168
<b>13.</b>	<b>INNE .....</b>	<b>170</b>

# 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:

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

### **1.9.      *Oszacowanie kosztów***

Projekt zrealizuje zespół czteroosobowy. W skład zespołu wchodzi:

- Projekt Manager (Łukasz Janus),
- Developerzy, Testerzy (Łukasz Witek vel Witkowski, Mateusz Marchelewicz),
- Tester, QA (Bartłomiej Bukowski).

Przewidywany czas trwania projektu:

- start 3 kwietnia 2017,
- koniec 17 czerwca 2017.

Praca wykonywana będzie regularnie w biurze w Łodzi lub, w uzasadnionych przypadkach, zdalnie, w godzinach 8:00-16:00, w dni robocze. W zaistnieniu wymogu konsultacji, w wybrane soboty jest przewidziana możliwość spotykania się zespołu w sprawie projektu. Spotkania będą mieć charakter nieformalnych i nie będą wchodzić w skład kosztów projektu.

Stawki godzinowe wynoszą:

30zł/h – developerzy,  
25zł/h – Projekt Manager,  
20zł/h – Tester&QA.

Projekt nie wymaga dodatkowych nakładów w postaci zakupu programów i sprzętu.

<b>Kalendarz pracy:</b>	<b>Ilość godzin:</b>	<b>Dni pracujące:</b>	<b>Dni wolne (święta, weekendy):</b>
Kwiecień	152	19	11
Maj	168	21	10
Czerwiec	88	11	9
<b>razem godzin:</b>	<b>408</b>		
Koszty osobowe:			
Projekt Manager	25złh	10 200 zł	
Developerzy	30zł/h	24 480 zł	
Tester &QA	20zł/h	8 160 zł	
<b>Razem osobowe:</b>		<b>42 840 zł</b>	
Koszty pozostałe (najem lokalu z dostępem do sieci, media, prąd)	1000zł/mies	2 548 zł	
<b>Razem całość:</b>		<b>45 388 zł</b>	

#### **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 |

- opis wymaganych zasobów:

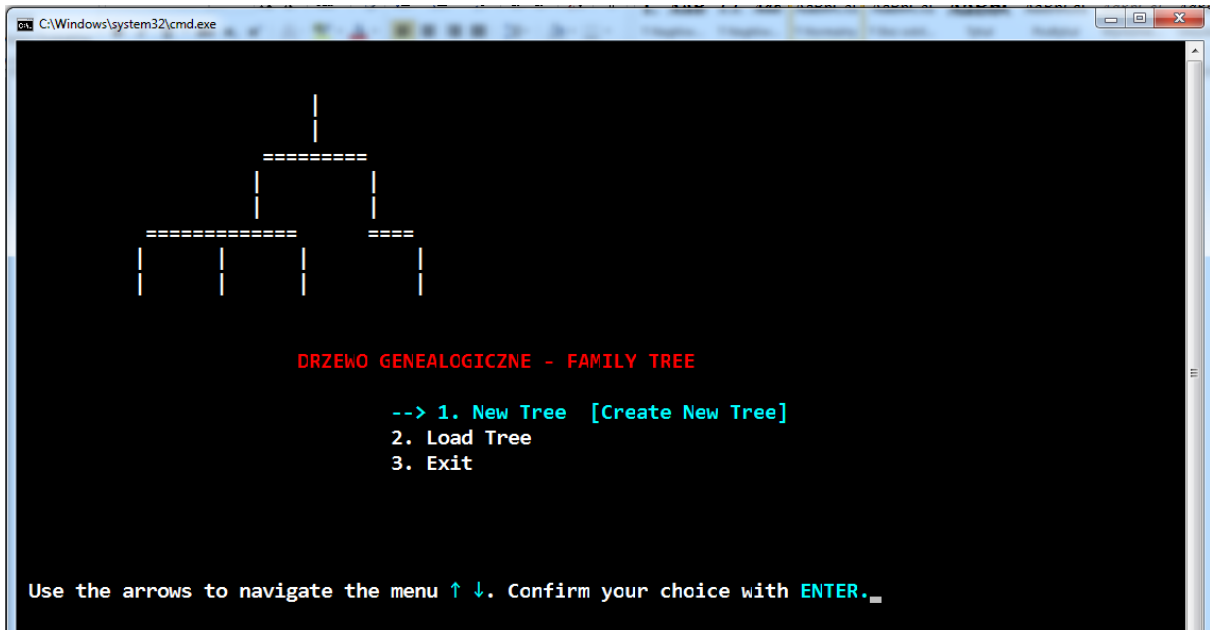
**pracownicy** – Łukasz Janus (kierownik grupy), Mateusz Marchelewicz, Łukasz Witek, Bartosz Bukowski

**oprogramowanie** – Visual Studio 2015 z wtyczką do GitHuba, LocMetrics, CppDepend, Gantt Project

**sprzęt** – komputery z systemem Windows oraz zainstalowanym w/w Visual Studio 15 i programem do liczenia kodu LocMetrics

## 1.11. Proponowana wizualizacja programu

a) aplikacja konsolowa (przykładowe zrzuty z menu)



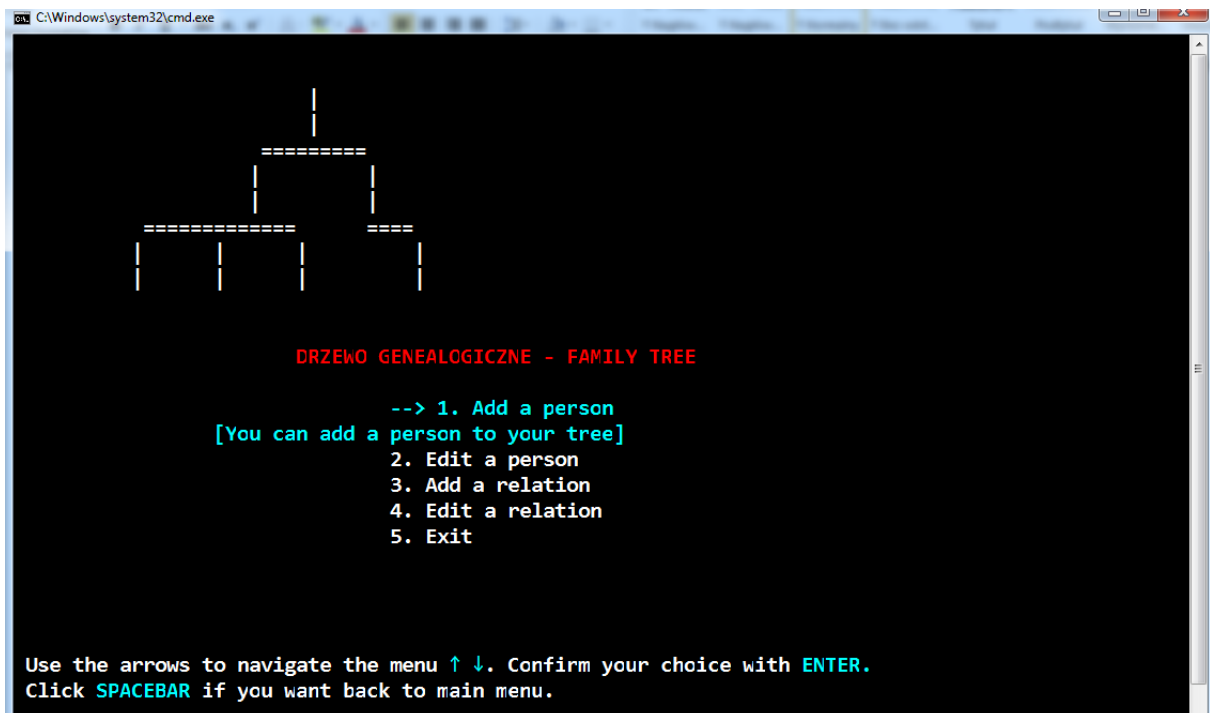
```
C:\Windows\system32\cmd.exe
```

```
      |
      |
=====
      |      |
=====  |  =====
|         |         |
|         |         |
|         |         |

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
```

```
      |
      |
=====
      |      |
=====  |  =====
|         |         |
|         |         |
|         |         |

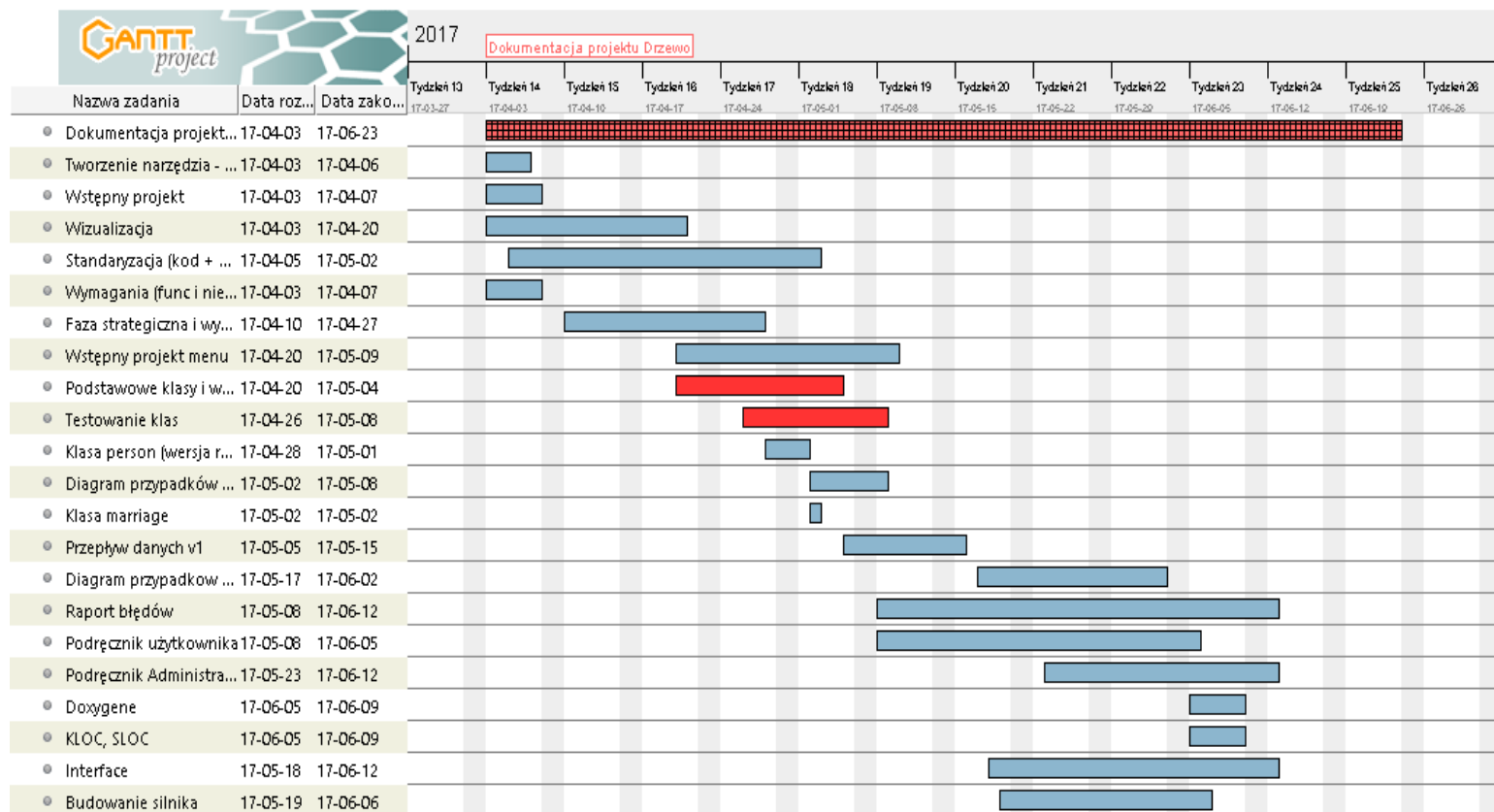
DRZEWO GENEALOGICZNE - FAMILY TREE

--> 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.
```



## 1.12 Diagram Gantt'a (stan na 12.06.2017)



## 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 nieznanych 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),
- 300 MB HDD,
- urządzenie graficzne z obsługą programu DirectX 9.

#### **Wydajność:**

- liczba transakcji obsłużonych w ciągu sekundy
- czas odpowiedzi (ok. 1-2 s),
- szybkość odświeżania ekranu (ok. 1-2 s.)

#### **Łatwość użytkowania:**

- liczba stron dokumentacji (około 190 stron, dok. Firmy + dok. Projektu)

## **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:

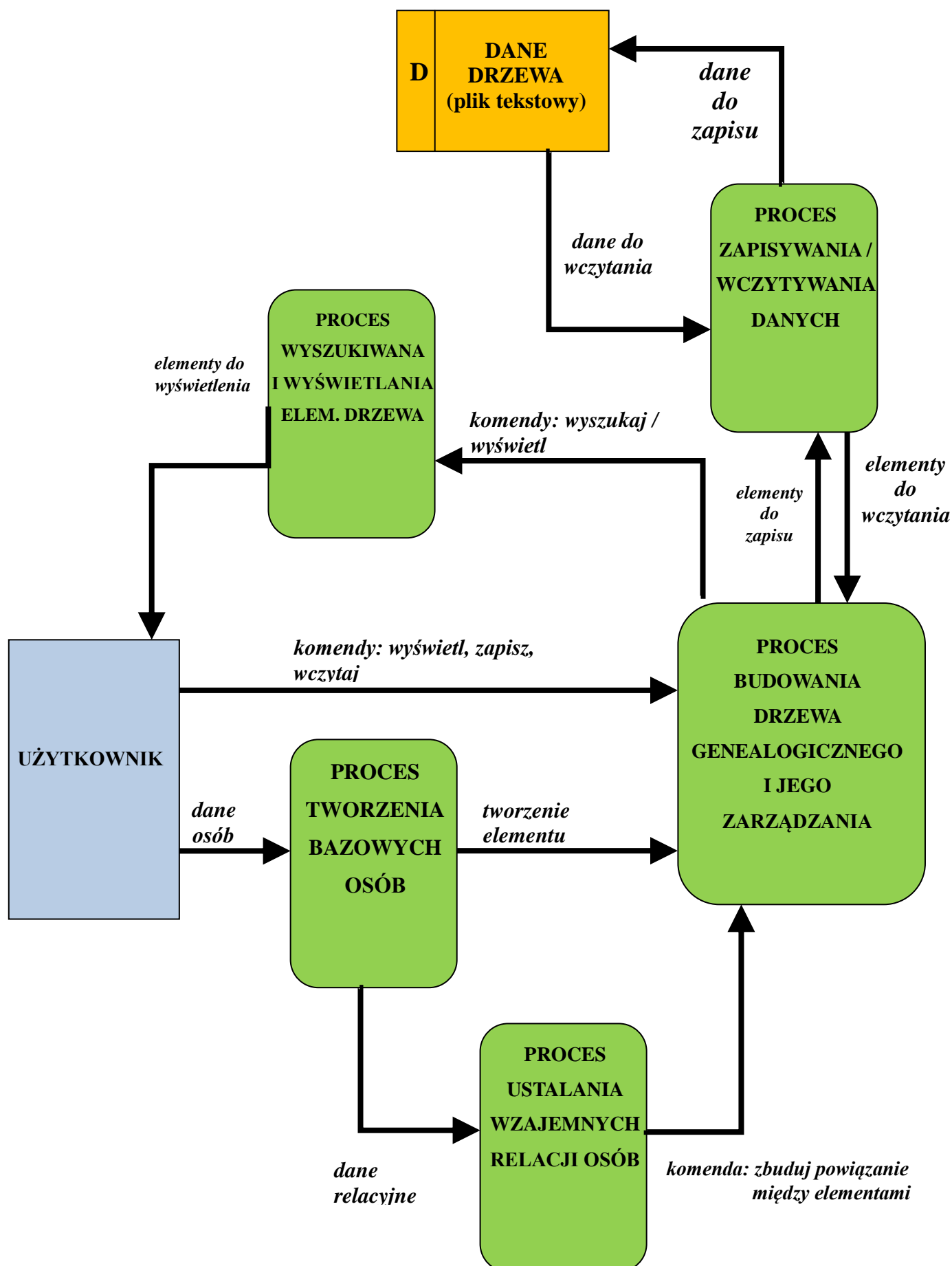
- **funkcje systemu** (zostały uzgodnione z klientem),
- **wydajność systemu** (przeszukiwanie bazy 5000 pozycji w ciągu  $< 1$  sek),
- **interfejsy zewnętrzne** (okienkowy),
- **wykonywane operacje** (dokładnie wyspecyfikowane),
- **wymagane zasoby sprzętowe** (określone),
- **sposoby weryfikacji** (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

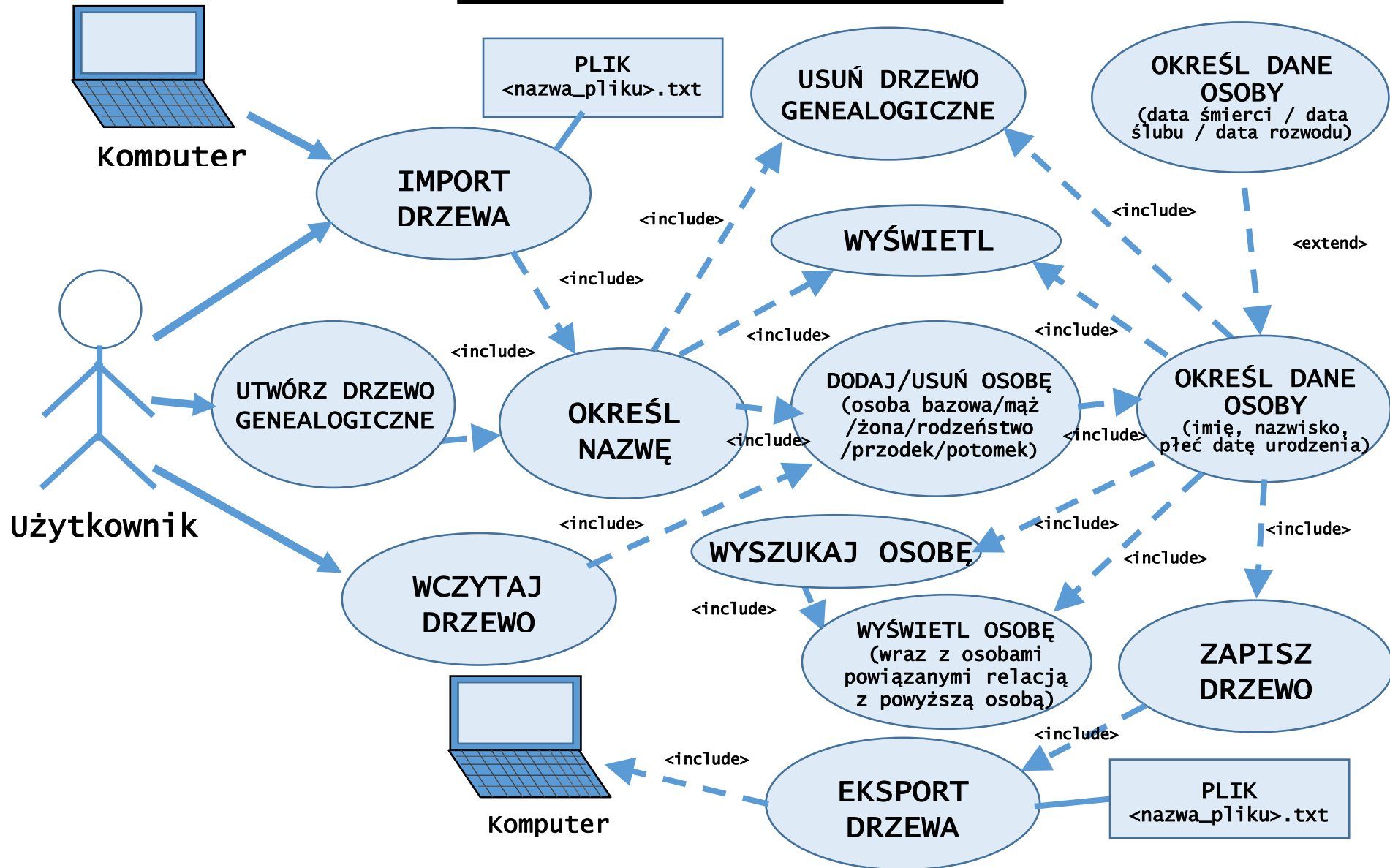
- k) u: wczytuje drzewo i powtarza np. punkty c-j
- alternatywny ciąg zdarzeń,
  - a) u: użytkownik rezygnuje z drzewa i je usuwa
  - b) u: określa dodatkowe dane (np. datę śmierci, ślubu, rozwodu) lub usuwa osobę
  - c) u: określa dodatkowe dane (np. datę śmierci, ślubu, rozwodu) lub usuwa osobę
  - 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



3.2. Diagram przypadków użycia

**DIAGRAM PRZYPADKÓW UŻYCIA**





## 4. DOKUMENTACJA FAZY PROJEKTOWANIA

### 4.1. *Lista klas i powiązania między klasami*

Klasa <b>C_date</b>	klasa potomna po C_day, C_month, C_year
Klasa <b>C_day</b>	klasa bazowa dla C_date
Klasa <b>C_month</b>	klasa bazowa dla C_date
Klasa <b>C_year</b>	klasa bazowa dla C_date

Klasa <b>C_data</b>	klasa podstawowa, bazowa dla C_first_name, C_last_name, C_gender, C_id
Klasa <b>C_first_name</b>	klasa pochodna, która dziedziczy po C_data
Klasa <b>C_last_name</b>	klasa pochodna, która dziedziczy po C_data
Klasa <b>C_gender</b>	klasa pochodna, która dziedziczy po C_data
Klasa <b>C_id</b>	klasa pochodna, która dziedziczy po C_data

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

Klasa <b>C_grandchildren</b>	klasa pochodna, która dziedziczy po C_relation
Klasa <b>C_sibling</b>	klasa pochodna, która dziedziczy po C_relation

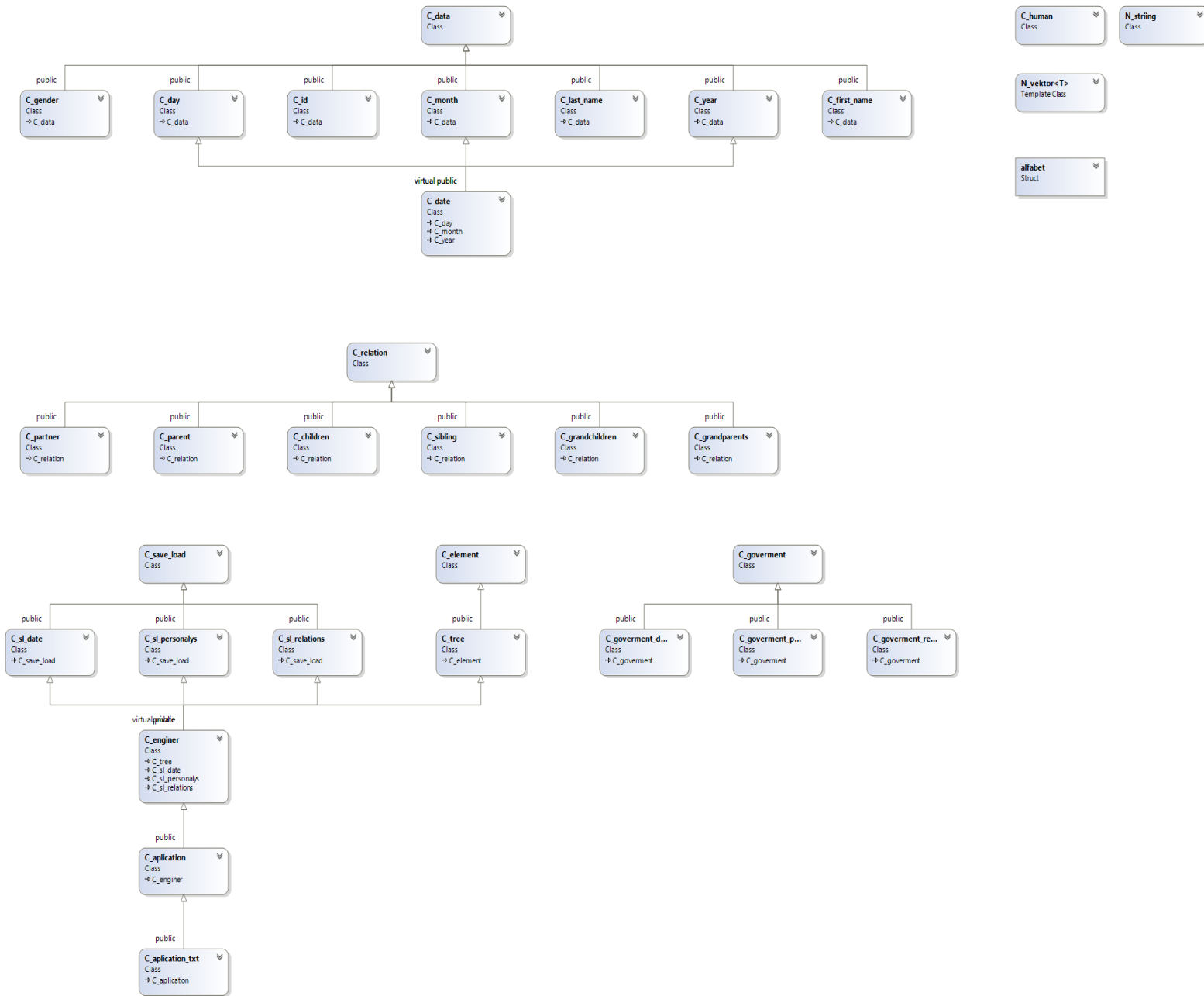
Klasa <b>C_element</b> - klasa podstawowa, bazowa dla C_tree, pochodna dla C_human,
Klasa <b>C_tree</b> - klasa podstawowa, dziedziczy i dzięki której działa C_engineer,
Klasa <b>C_human</b> – klasa podstawowa, bazowa dla C_first_name, C_last_name, C_id, C_date,
Klasa <b>C_government</b> - klasa bazowa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

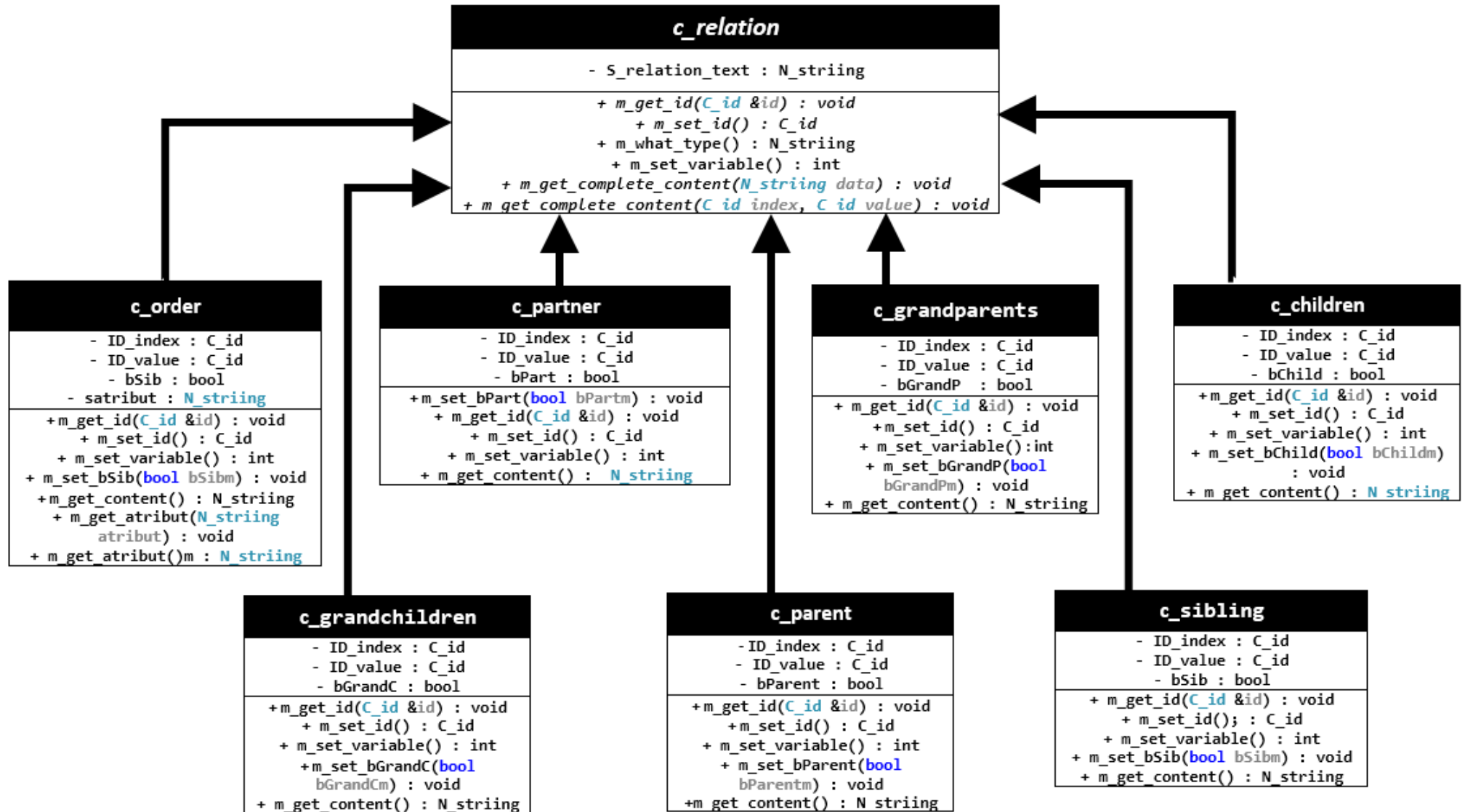
Klasa <b>C_engineer</b> - klasa podstawowa, bazowa dla C_tree, C_sl_date, C_sl_personalys,
Klasa <b>C_save_load</b> – klasa bazowa dla innych klas do wczytywania danych
Klasa <b>C_sl_date</b> – klasa dziedzicząca po C_save_load
Klasa <b>C_sl_personalys</b> – klasa dziedzicząca po C_save_load
Klasa <b>C_sl_relations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

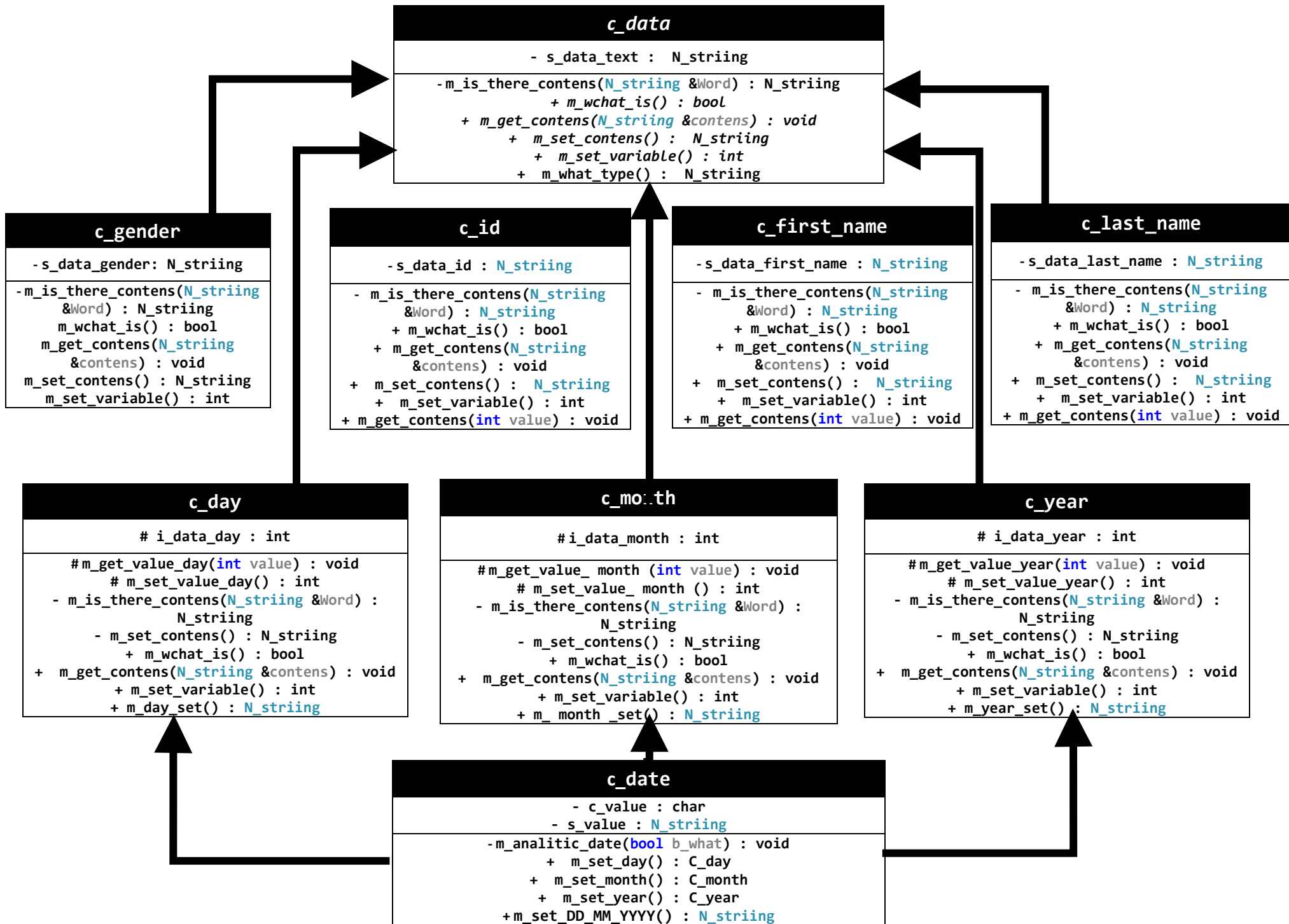
Klasa <b>C_application</b> - Klasa czysto abstrakcyjna dla klas poświęconych interfejsowi, klasa dziecko po klasie engineer
Klasa <b>C_application_txt</b> – klasa dziedzicząca po C_application

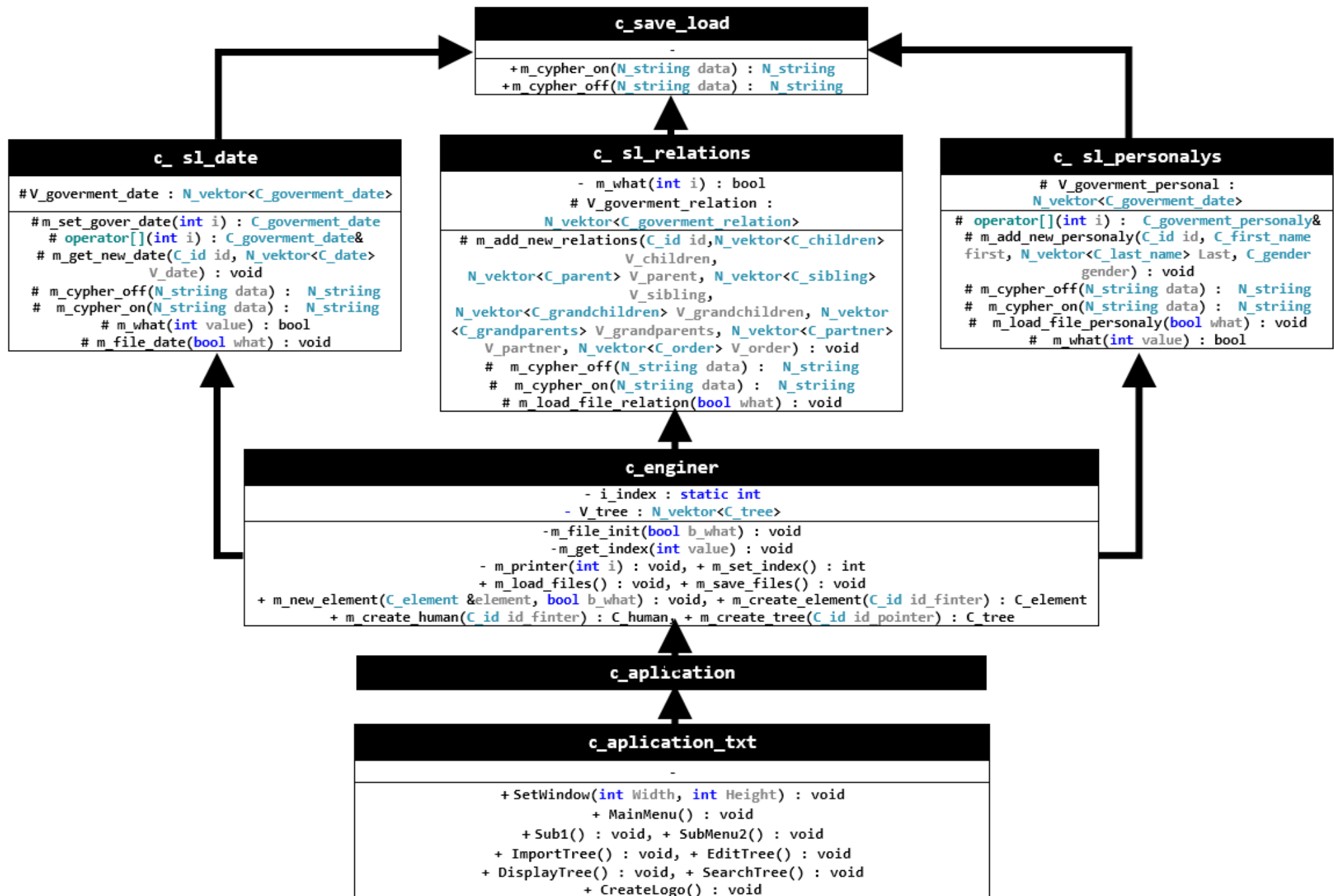
Klasa <b>N_striing</b> – klasa podstawowa, bazowa (zastępuje bibliotekę String)
Klasa <b>N_vektor</b> – klasa podstawowa, bazowa (zastępuje bibliotekę Vector)

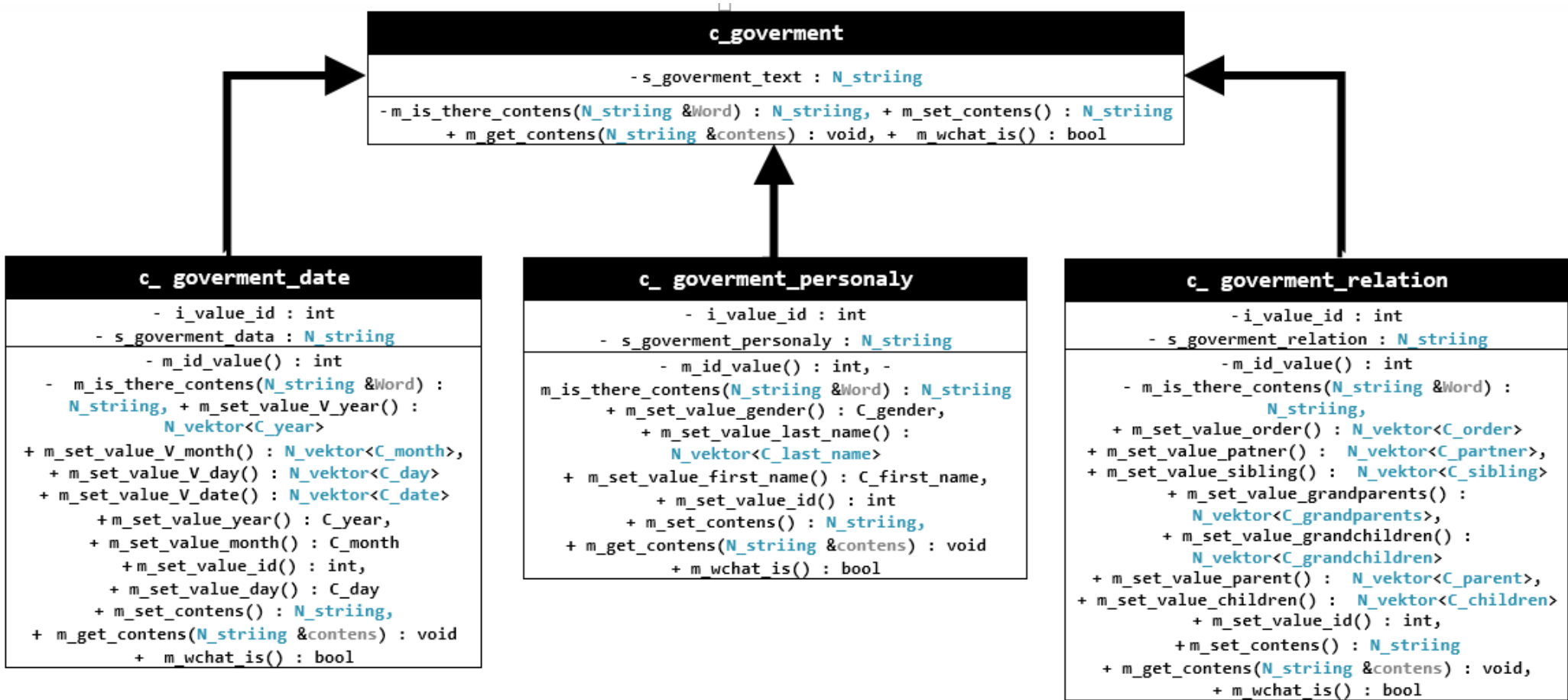
## 4.2. Diagram powiązań między klasami











c_element	
<pre> - V_children : N_vektor&lt;C_children&gt; - V_parent : N_vektor&lt;C_parent&gt; - V_sibling : N_vektor&lt;C_sibling&gt; - V_grandchildren : N_vektor&lt;C_grandchildren&gt; - V_grandparents : N_vektor &lt;C_grandparents&gt; - V_partner : N_vektor &lt;C_partner&gt; - V_order : N_vektor &lt;C_order&gt; # Human : C_human </pre>	<pre> +m_set_sibling(int value) : C_sibling +m_set_grandchildren(int value) : C_grandchildren +m_set_grandparents(int value) : C_grandparents +m_set_partner(int value) : C_partner +m_set_order(int value) : C_order +&amp; m_clean() : C_element +m_clean_children(): void +m_clean_parent(): void +m_clean_sibling(): void +m_clean_grandparents(): void +m_clean_grandchildren(): void +m_clean_partner(): void +m_clean_order(): void +m_delete_children(): void +m_delete_parent(): void +m_delete_sibling(): void +m_delete_partner(): void +m_delete_order(): void +m_delete_children(int value) : void +m_delete_parent(int value) : void +m_delete_sibling(int value) : void +m_delete_grandchildren(int value) : void +m_delete_grandparents(int value) : void +m_delete_partner(int value) : void +m_delete_order(int value): void +m_set_v_grandparents() : N_vektor&lt;C_grandparents&gt; +m_set_v_grandchildren() : N_vektor&lt;C_grandchildren&gt; +m_set_v_parent() : N_vektor&lt;C_parent&gt; +m_set_v_children() : N_vektor&lt;C_children&gt; +m_set_v_partner() : N_vektor &lt;C_partner&gt; +m_set_v_sibling() :N_vektor&lt;C_sibling&gt; +m_set_v_order() : N_vektor &lt;C_order&gt; </pre>
<pre> +m_get_children(C_children &amp;children) : void + m_get_parent(C_parent &amp;parent) : void +m_get_sibling(C_sibling &amp;sibling) : void +m_get_grandchildren(C_grandchildren &amp;grandchildren) : void +m_get_grandparents(C_grandparents &amp;grandparents) : void +m_get_partner(C_partner &amp;partner) : void +m_get_order(C_order &amp;order) : void +m_update_children(int value,C_children &amp;children) : void +m_update_parent(int value,C_parent &amp;parent) : void +m_update_sibling(int value,C_sibling &amp;sibling) : void +m_update_human(const C_human &amp;human) : void m_update_grandchildren(int value, C_grandchildren &amp;human) : void +m_update_grandparents(int value, C_grandparents &amp;human) : void +m_update_partner(int value, C_partner &amp;partner) : void +m_update_order(int value, C_order &amp;order) : void +m_set_Human() : C_human +m_set_children() : C_children +m_set_parent() : C_parent +m_set_sibling() : C_sibling +m_set_partner() : C_partner +m_set_grandchildren() : C_grandchildren +m_set_grandparents() : C_grandparents +m_set_order() : C_order +m_set_children(int value) : C_children +m set parent(int value) : C parent </pre>	

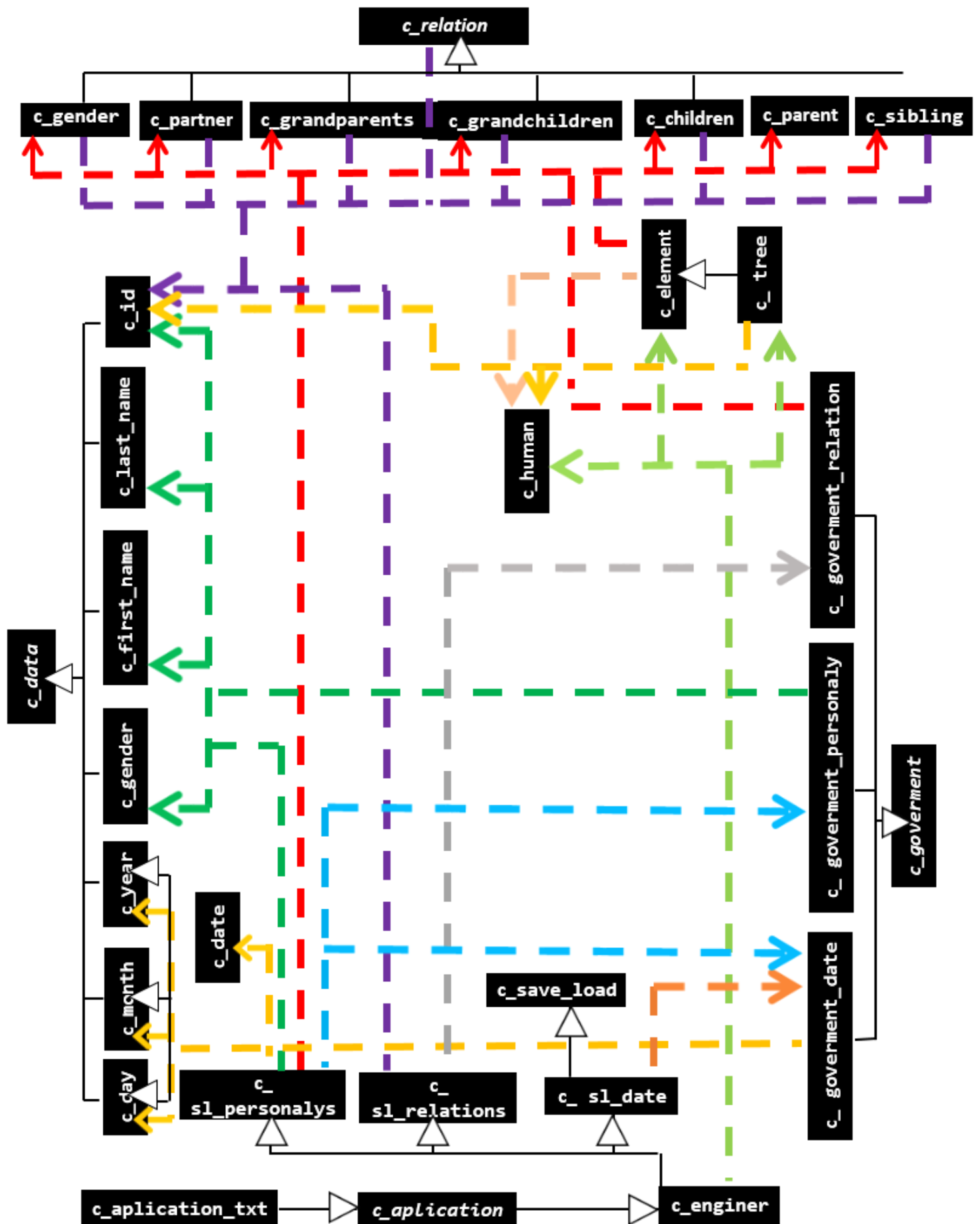


c_tree
<pre> - V_human_grandparent : N_vektor&lt;C_human&gt;, - V_human_grandchildren : N_vektor&lt;C_human&gt; - V_human_children : N_vektor&lt;C_human&gt;,- V_human_parent : N_vektor&lt;C_human&gt; - V_human_sibling : N_vektor&lt;C_human&gt;,- V_human_partner : N_vektor&lt;C_human&gt; - V_human_order : N vektor&lt;C human&gt; </pre>
<pre> +m_add_human(C_human &amp;h, int &amp;data) : void, + m_update_human(C_human &amp;h, int &amp;data, int ivalue) : void m_delete_human(C_human &amp;h, int &amp;data, int ivalue) : void, + m_delete_human(C_human &amp;h, int &amp;data) : void + m_add_id(const C_id &amp;id) : void, + m_get_id() : C_id + m_get_human(int &amp;data, int ivalue) : C_id + m_get_index_human() : C_human </pre>



c_human	
<pre> - V_date : N_vektor&lt;C_date&gt; - C_id Id : C_id - First : C_first_name - V_last : N_vektor&lt;C_last_name&gt; - Gender : C_gender </pre>	
<pre> +m_get_first_name(C_first_name &amp;f_name) : void +m_get_first_name(N_striing &amp;f_name) : void +m_get_last_name(C_last_name &amp;l_name) : void +m_get_last_name(N_striing &amp;l_name) : void +m_get_gender(C_gender &amp;gender) : void +m_get_gender(N_striing &amp;gender) : void +m_get_gender(bool gender) : void +m_shift_id(N_striing &amp;id) : void +m_shift_id(int id) : void +m_shift_id(C_id &amp;id) : void +m_get_date(C_date date) : void +m_delete_first_name() : void +m_delete_last_name(int value) : void +m_delete_last_name() : void +m_delete_gender() : void +m_delete_date(int value) : void +m_delete_date() : void +m_update_date(int value, C_date&amp; date) : void +m_update_last_name(int value, C_last_name&amp; l_name) : void +m_update_last_name(int value, N_striing&amp; l_name) : void +interf_cut(N_striing &amp;first, N_striing &amp;last, C_human &amp;human,             int cut) : void +interf_m(C_human &amp;human, C_date &amp;d, C_date ds = NULL) : void +interf_mb(N_striing firstname, N_striing lastname, C_date &amp;du, C_date ds = NULL, char poz = '*', char pion = ' ') : void interf_mbd(N_striing firstname, N_striing lastname, C_date &amp;du, +C date ds = NULL, char poz = '*', char pion = ' ') : void </pre>	<pre> +m_short_interface_personaly() : N_striing +m_short_interface_date() : N_striing +m_clear() : C_human&amp; +m_clear_date() : C_human&amp; +m_clear_last_name() : C_human&amp; +m_set_first_name() : C_first_name +m_set_last_name() : C_last_name +m_set_last_name(int value) : C_last_name +m_set_gender() : C_gender +m_set_id() : C_id +m_set_date(int value) : C_date +m_set_date() : C_date +m_set_Vdate() : N_vektor&lt;C_date&gt; +m_set_V_last_name() : N_vektor&lt;C_last_name&gt; </pre>

N_striing	
<pre> - Table : char* - Size : int </pre>	
<pre> const char* m_c_str() m_itoa(long long i) : N_striing&amp; m_itoa(long long i) : N_striing&amp; m_atoi(int variable_start, int variable_stop): long long m_push_back(const char &amp;Gover) : N_striing&amp; m_push_back(const char Gover[]) : N_striing&amp; m_push_front(const char &amp;Gover) : N_striing&amp; m_push_front(const char Gover[]) : N_striing&amp; m_insert(int value,const char Gover) : N_striing&amp; m_erase_ray(int value_front, int value_back): N_striing&amp; </pre>	<pre> m_erase_ray(int value_front): N_striing&amp; m_shift(int i, const char &amp;value) : N_striing&amp; m_insert(int value, const char Gover[]) : N_striing&amp; m_swap(const char &amp;Gover_old, const char &amp;Gover_new) : N_striing&amp; m_swap(const char Gover_old[], const char Gover_new[]) : N_striing&amp; m_pop_back() : N_striing&amp;, m_pop_front() : N_striing&amp; m_clear() : N_striing, m_erase(int i) : N_striing&amp; m_cut(int value_front, int value_back) : N_striing m_wchat_char(const char &amp;variable) : bool m_wchat_char(const char variable[]) : bool m_cut(int value_front) : N_striing m_size() : int m_getline(std::ifstream &amp;is) : N_striing&amp; </pre>



## 5. DOKUMENTACJA FAZY IMPLEMENTACJI

### 5.1. Policzony kod dla modułów w projekcie FamilyTree

(LOC, SLOC, McCabe, KDSI):

#### KDSI:

ręcznie:	source instruction (if, for, switch)	
main	12	+1 ('main')
element.cpp	4	
goverment.cpp	5	
goverment_date.cpp	38	
goverment_personaly.cpp	25	
goverment_relation.cpp	43	
human.cpp	63	
tree.cpp	44	
date.cpp	22	
year.cpp	6	
month.cpp	7	
day.cpp	7	
sl_personalys.cpp	20	
sl_date.cpp	20	
save_load.cpp	50	
enginer.cpp	36	
alphabet.h	3	
sl_relations.cpp	28	
aplication.cpp	5	
aplication.txt.cpp	96	
Vektor.h	27	
Striing.cpp	94	
data.cpp	4	
last_name.cpp	10	
id.cpp	9	
gender.cpp	6	
first_name.cpp	9	

sibling.cpp	11	
relation.cpp	4	
partner.cpp	11	
parent.cpp	11	
order.cpp	10	
grantparents.cpp	11	
grantchildren.cpp	11	
children.cpp	11	
data.cpp	4	(bazowa)
id.cpp	4	(bazowa)
first_name.cpp	4	(bazowa)
Striing.cpp	81	(bazowa)
<b>razem</b>	<b>867</b>	
<b>DSI</b>	<b>1494,9</b>	
<b>KDSI</b>	<b>1,4949</b>	

Checkpoint Name	Baseline
<b>Created On</b>	12 Jun 2017
<b>Files</b>	64
<b>Lines</b>	7233
<b>Statements</b>	4806
<b>% Branches</b>	21,5
<b>% Comments</b>	17,1
<b>Class Defs</b>	31
<b>Methods/Class</b>	16,48
<b>Avg Stmt/Method</b>	5,1
<b>Max Complexity</b>	98
<b>Max Depth</b>	8
<b>Avg Depth</b>	1,64
<b>Avg Complexity</b>	2,77
<b>Functions</b>	16

## (LOC, SLOC, McCabe):

### 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		
McCabe VG Complexity	43		

## Folder Data / Date (**date.h, date.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	32
Directories	1	CLOC, Comment Lines	22
LOC, Lines of Code	254	CWORD, Comment Words	247
BLOC, Blank Lines	3	HCLOC, Header Comments	15
SLOC-P, Executable Physical	229	HCWORD, Header Words	47
SLOC-L, Executable Logical	155		
McCabe VG Complexity	30		

## Folder Data / Date (**day.h, day.cpp**)

locmetrics.com			
Progress			
Source Files	2	C&SLOC, Code & Comment	9
Directories	1	CLOC, Comment Lines	19
LOC, Lines of Code	102	CWORD, Comment Words	104
BLOC, Blank Lines	6	HCLOC, Header Comments	19
SLOC-P, Executable Physical	77	HCWORD, Header Words	81
SLOC-L, Executable Logical	57		
McCabe VG Complexity	14		

## Folder Data / Date (**month.h**, **month.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	19
LOC, Lines of Code	101	CWORD, Comment Words	82
BLOC, Blank Lines	5	HCLOC, Header Comments	19
SLOC-P, Executable Physical	77	HCWORD, Header Words	81
SLOC-L, Executable Logical	57		
McCabe VG Complexity	14		

## Folder Data / Date (**year.h**, **year.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	20
LOC, Lines of Code	102	CWORD, Comment Words	82
BLOC, Blank Lines	7	HCLOC, Header Comments	20
SLOC-P, Executable Physical	75	HCWORD, Header Words	81
SLOC-L, Executable Logical	56		
McCabe VG Complexity	12		

## Folder Data / Personalys (**data.h**, **data.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	22
LOC, Lines of Code	67	CWORD, Comment Words	87
BLOC, Blank Lines	0	HCLOC, Header Comments	17
SLOC-P, Executable Physical	45	HCWORD, Header Words	65
SLOC-L, Executable Logical	32		
McCabe VG Complexity	7		

### Folder Data / Personalys (**first\_name.h, first\_name.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	18
LOC, Lines of Code	108	CWORD, Comment Words	67
BLOC, Blank Lines	6	HCLOC, Header Comments	17
SLOC-P, Executable Physical	84	HCWORD, Header Words	65
SLOC-L, Executable Logical	61		
McCabe VG Complexity	18		

### Folder Data / Personalys (**gender.h, gender.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	17
LOC, Lines of Code	88	CWORD, Comment Words	65
BLOC, Blank Lines	7	HCLOC, Header Comments	17
SLOC-P, Executable Physical	64	HCWORD, Header Words	64
SLOC-L, Executable Logical	48		
McCabe VG Complexity	12		

### Folder Data / Personalys (**id.h, id.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	19
LOC, Lines of Code	98	CWORD, Comment Words	84
BLOC, Blank Lines	1	HCLOC, Header Comments	19
SLOC-P, Executable Physical	78	HCWORD, Header Words	83
SLOC-L, Executable Logical	58		
McCabe VG Complexity	16		



### Folder Data / Personalys (**last\_name.h, last\_name.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	17
LOC, Lines of Code	114	CWORD, Comment Words	64
BLOC, Blank Lines	11	HCLOC, Header Comments	17
SLOC-P, Executable Physical	86	HCWORD, Header Words	63
SLOC-L, Executable Logical	61		
McCabe VG Complexity	18		

### Folder Data / Relations (**children.h, children.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	3
Directories	1	CLOC, Comment Lines	18
LOC, Lines of Code	135	CWORD, Comment Words	85
BLOC, Blank Lines	5	HCLOC, Header Comments	18
SLOC-P, Executable Physical	112	HCWORD, Header Words	78
SLOC-L, Executable Logical	80		
McCabe VG Complexity	18		

### Folder Data / Relations (**grandchildren.h, grandchildren.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	17
LOC, Lines of Code	135	CWORD, Comment Words	71
BLOC, Blank Lines	5	HCLOC, Header Comments	17
SLOC-P, Executable Physical	113	HCWORD, Header Words	68
SLOC-L, Executable Logical	81		
McCabe VG Complexity	18		

### Folder Data / Relations (**grandparents.h**, **grandparents.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	17
LOC, Lines of Code	135	CWORD, Comment Words	71
BLOC, Blank Lines	5	HCLOC, Header Comments	17
SLOC-P, Executable Physical	113	HCWORD, Header Words	68
SLOC-L, Executable Logical	81		
McCabe VG Complexity	18		

### Folder Data / Relations (**order.h**, **order.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	24
Directories	1	CLOC, Comment Lines	18
LOC, Lines of Code	144	CWORD, Comment Words	159
BLOC, Blank Lines	5	HCLOC, Header Comments	18
SLOC-P, Executable Physical	121	HCWORD, Header Words	78
SLOC-L, Executable Logical	86		
McCabe VG Complexity	17		

### Folder Data / Relations (**parent.h**, **parent.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	18
LOC, Lines of Code	136	CWORD, Comment Words	81
BLOC, Blank Lines	4	HCLOC, Header Comments	18
SLOC-P, Executable Physical	114	HCWORD, Header Words	78
SLOC-L, Executable Logical	81		
McCabe VG Complexity	18		

### Folder Data / Relations (**partner.h**, **partner.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	14
LOC, Lines of Code	130	CWORD, Comment Words	42
BLOC, Blank Lines	4	HCLOC, Header Comments	14
SLOC-P, Executable Physical	112	HCWORD, Header Words	39
SLOC-L, Executable Logical	81		
McCabe VG Complexity	18		

### Folder Data / Relations (**relation.h**, **relation.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	15
LOC, Lines of Code	63	CWORD, Comment Words	48
BLOC, Blank Lines	1	HCLOC, Header Comments	15
SLOC-P, Executable Physical	47	HCWORD, Header Words	46
SLOC-L, Executable Logical	33		
McCabe VG Complexity	7		

### Folder Data / Relations (**sibling.h**, **sibling.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	20
Directories	1	CLOC, Comment Lines	18
LOC, Lines of Code	135	CWORD, Comment Words	146
BLOC, Blank Lines	5	HCLOC, Header Comments	18
SLOC-P, Executable Physical	112	HCWORD, Header Words	78
SLOC-L, Executable Logical	80		
McCabe VG Complexity	18		

## Folder Databases (**element.h**, **element.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	16
LOC, Lines of Code	214	CWORD, Comment Words	61
BLOC, Blank Lines	6	HCLOC, Header Comments	15
SLOC-P, Executable Physical	192	HCWORD, Header Words	50
SLOC-L, Executable Logical	160		
McCabe VG Complexity	7		

## Folder Databases (**government.h**, **government.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	14
LOC, Lines of Code	63	CWORD, Comment Words	41
BLOC, Blank Lines	7	HCLOC, Header Comments	14
SLOC-P, Executable Physical	42	HCWORD, Header Words	40
SLOC-L, Executable Logical	28		
McCabe VG Complexity	8		

## Folder Databases (**government\_date.h**, **government\_date.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	31
Directories	1	CLOC, Comment Lines	19
LOC, Lines of Code	309	CWORD, Comment Words	234
BLOC, Blank Lines	4	HCLOC, Header Comments	0
SLOC-P, Executable Physical	286	HCWORD, Header Words	1
SLOC-L, Executable Logical	200		
McCabe VG Complexity	61		

## Folder Databases (**government\_personaly.h**, **government\_personaly.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	26
Directories	1	CLOC, Comment Lines	21
LOC, Lines of Code	217	CWORD, Comment Words	242
BLOC, Blank Lines	4	HCLOC, Header Comments	0
SLOC-P, Executable Physical	192	HCWORD, Header Words	1
SLOC-L, Executable Logical	127		
McCabe VG Complexity	41		

### Folder Databases (**government\_relation.h**, **government\_relation.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	30
Directories	1	CLOC, Comment Lines	25
LOC, Lines of Code	361	CWORD, Comment Words	306
BLOC, Blank Lines	3	HCLOC, Header Comments	0
SLOC-P, Executable Physical	333	HCWORD, Header Words	1
SLOC-L, Executable Logical	237		
McCabe VG Complexity	66		

### Folder Databases (**human.h**, **human.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	21
LOC, Lines of Code	599	CWORD, Comment Words	88
BLOC, Blank Lines	38	HCLOC, Header Comments	19
SLOC-P, Executable Physical	540	HCWORD, Header Words	77
SLOC-L, Executable Logical	350		
McCabe VG Complexity	72		

### Folder Databases (**tree.h**, **tree.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	2
Directories	1	CLOC, Comment Lines	21
LOC, Lines of Code	789	CWORD, Comment Words	66
BLOC, Blank Lines	7	HCLOC, Header Comments	15
SLOC-P, Executable Physical	761	HCWORD, Header Words	50
SLOC-L, Executable Logical	435		
McCabe VG Complexity	174		

## Folder Engineer (**alphabet.h**)

Progress			
Source Files	1	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	0
LOC, Lines of Code	30	CWORD, Comment Words	1
BLOC, Blank Lines	1	HCLOC, Header Comments	0
SLOC-P, Executable Physical	29	HCWORD, Header Words	0
SLOC-L, Executable Logical	18		
McCabe VG Complexity	3		

## Folder Engineer (**engineer.h, engineer.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	3
Directories	1	CLOC, Comment Lines	20
LOC, Lines of Code	268	CWORD, Comment Words	68
BLOC, Blank Lines	2	HCLOC, Header Comments	14
SLOC-P, Executable Physical	246	HCWORD, Header Words	41
SLOC-L, Executable Logical	163		
McCabe VG Complexity	47		

## Folder Engineer (**save\_load.h, save\_load.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	3
Directories	1	CLOC, Comment Lines	373
LOC, Lines of Code	413	CWORD, Comment Words	987
BLOC, Blank Lines	1	HCLOC, Header Comments	13
SLOC-P, Executable Physical	39	HCWORD, Header Words	30
SLOC-L, Executable Logical	24		
McCabe VG Complexity	8		

## Folder Engineer (**sl\_date.h, sl\_date.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	15
Directories	1	CLOC, Comment Lines	22
LOC, Lines of Code	185	CWORD, Comment Words	100
BLOC, Blank Lines	8	HCLOC, Header Comments	13
SLOC-P, Executable Physical	155	HCWORD, Header Words	28
SLOC-L, Executable Logical	111		
McCabe VG Complexity	27		

### Folder Engineer (**sl\_personalys.h**, **sl\_personalys.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	15
Directories	1	CLOC, Comment Lines	20
LOC, Lines of Code	172	CWORD, Comment Words	91
BLOC, Blank Lines	4	HCLOC, Header Comments	13
SLOC-P, Executable Physical	148	HCWORD, Header Words	28
SLOC-L, Executable Logical	105		
McCabe VG Complexity	25		

### Folder Engineer (**sl\_relations.h**, **sl\_relations.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	14
Directories	1	CLOC, Comment Lines	17
LOC, Lines of Code	224	CWORD, Comment Words	76
BLOC, Blank Lines	4	HCLOC, Header Comments	13
SLOC-P, Executable Physical	203	HCWORD, Header Words	28
SLOC-L, Executable Logical	138		
McCabe VG Complexity	34		

### Folder Interface (**aplication.h**, **aplication.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	1
Directories	1	CLOC, Comment Lines	17
LOC, Lines of Code	56	CWORD, Comment Words	45
BLOC, Blank Lines	5	HCLOC, Header Comments	13
SLOC-P, Executable Physical	34	HCWORD, Header Words	35
SLOC-L, Executable Logical	22		
McCabe VG Complexity	8		

### Folder Interface (**aplication\_txt.h**, **aplication\_txt.cpp**)

Progress			
Source Files	2	C&SLOC, Code & Comment	80
Directories	1	CLOC, Comment Lines	240
LOC, Lines of Code	1040	CWORD, Comment Words	1309
BLOC, Blank Lines	140	HCLOC, Header Comments	18
SLOC-P, Executable Physical	660	HCWORD, Header Words	65
SLOC-L, Executable Logical	461		
McCabe VG Complexity	90		

## 5.2. Policzone osobno klasy w projekcie FamilyTree (alfabetycznie):

### Structure Info

public struct **alfabet**

- Drzewo\_genealogiczne

# Issues (Cumulated): 1  
Debt Rating: **A**  
All Debt: 0min 0s  
All Annual Interest: 0min 0s  
Breaking Point: 0 milli-second  
  
# lines of code (LOC): 14  
# lines of comment: 0  
Percentage Comment: 0%  
Cyclomatic Complexity (CC): 16  
# Methods: 9  
# Fields: 3  
Type Level: 1  
Difficulty Level: 16  
Implementation Time(in seconds): 567  
Estimated Delivered Bugs: 0,16  
Depth of inheritance: 0  
# Children: 0  
# Types used: 1  
# Types using me: 0  
Association Between Types (ABT): 2  
Type Rank: 0.15  
Lack of Cohesion Of Methods (LCOM): -0.33  
LCOM Henderson-Sellers (LCOMHS): -0.5

### Class Info

public class **C\_aplication** : **C\_enginer**

- Drzewo\_genealogiczne

# Issues (Cumulated): 1  
Debt Rating: **B**  
All Debt: 9min  
All Annual Interest: 2min 0s  
Breaking Point: 4 years  
  
# lines of code (LOC): 13  
# lines of comment: 4  
Percentage Comment: 23,52941  
Cyclomatic Complexity (CC): 11  
# Methods: 6  
# Fields: 0  
Type Level: 9  
Difficulty Level: 10  
Implementation Time(in seconds): 126  
Estimated Delivered Bugs: 0,06  
Depth of inheritance: 3  
# Children: 1  
# Types used: 4  
# Types using me: 1  
Association Between Types (ABT): 6  
Type Rank: 0.22  
Lack of Cohesion Of Methods (LCOM): 0  
LCOM Henderson-Sellers (LCOMHS): 0



## Class Info

public class C\_aplication\_txt : C\_aplication

- Drzewo\_genealogiczne

# Issues (Cumulated): 14

Debt Rating: **C**

All Debt: 1d 0h

All Annual Interest: 1h 9min

Breaking Point: 7 years

# lines of code (LOC): 421

# lines of comment: 300

Percentage Comment: 41,60888

Cyclomatic Complexity (CC): 112

# Methods: 16

# Fields: 0

Type Level: 10

Difficulty Level: 44

Implementation Time(in seconds): 48 458

Estimated Delivered Bugs: 3,04

Depth of inheritance: 4

# Children: 0

# Types used: 9

# Types using me: 0

Association Between Types (ABT): 41

Type Rank: 0.15

Lack of Cohesion Of Methods (LCOM): 0

LCOM Henderson-Sellers (LCOMHS): 0

## Class Info

public class C\_children : C\_relation

- Drzewo\_genealogiczne

# Issues (Cumulated): 5

Debt Rating: **A**

All Debt: 10min

All Annual Interest: 1h 18min

Breaking Point: 47 days

# lines of code (LOC): 47

# lines of comment: 2

Percentage Comment: 4,081633

Cyclomatic Complexity (CC): 31

# Methods: 16

# Fields: 3

Type Level: 5

Difficulty Level: 33

Implementation Time(in seconds): 3 627

Estimated Delivered Bugs: 0,54

Depth of inheritance: 1

# Children: 0

# Types used: 3

# Types using me: 4

Association Between Types (ABT): 20

Type Rank: 0.36

Lack of Cohesion Of Methods (LCOM): 0.56

LCOM Henderson-Sellers (LCOMHS): 0.6

## Class Info

public abstract class C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 3

Debt Rating: **B**

All Debt: 22min

All Annual Interest: 22min

Breaking Point: 12 months

# lines of code (LOC): 14

# lines of comment: 3

Percentage Comment: 17,64706

Cyclomatic Complexity (CC): 16

# Methods: 12

# Fields: 1

Type Level: 2

Difficulty Level: 10

Implementation Time(in seconds): 226

Estimated Delivered Bugs: 0,09

Depth of inheritance: 0

# Children: 10

# Types used: 1

# Types using me: 10

Association Between Types (ABT): 4

Type Rank: 4.72

Lack of Cohesion Of Methods (LCOM): 0.58

LCOM Henderson-Sellers (LCOMHS): 0.64

## Class Info

public class C\_date : C\_day, C\_month, C\_year

- Drzewo\_genealogiczne

# Issues (Cumulated): 5

Debt Rating: **C**

All Debt: 2h 30min

All Annual Interest: 39min

Breaking Point: 3 years

# lines of code (LOC): 109

# lines of comment: 34

Percentage Comment: 23,77622

Cyclomatic Complexity (CC): 69

# Methods: 29

# Fields: 2

Type Level: 4

Difficulty Level: 40

Implementation Time(in seconds): 11 093

Estimated Delivered Bugs: 1,14

Depth of inheritance: 2

# Children: 0

# Types used: 4

# Types using me: 4

Association Between Types (ABT): 35

Type Rank: 0.4

Lack of Cohesion Of Methods (LCOM): 0.71

LCOM Henderson-Sellers (LCOMHS): 0.73

## Class Info

public class C\_day : C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 1

Debt Rating: **A**

All Debt: 6min

All Annual Interest: 20min

Breaking Point: 3 months

# lines of code (LOC): 36

# lines of comment: 8

Percentage Comment: 18,18182

Cyclomatic Complexity (CC): 26

# Methods: 17

# Fields: 1

Type Level: 3

Difficulty Level: 20

Implementation Time(in seconds): 1 686

Estimated Delivered Bugs: 0,32

Depth of inheritance: 1

# Children: 1

# Types used: 2

# Types using me: 4

Association Between Types (ABT): 11

Type Rank: 0.58

Lack of Cohesion Of Methods (LCOM): -0.12

LCOM Henderson-Sellers (LCOMHS): -0.12

## Class Info

public class C\_element

- Drzewo\_genealogiczne

# Issues (Cumulated): 6

Debt Rating: **C**

All Debt: 3h 15min

All Annual Interest: 1h 8min

Breaking Point: 34 months

# lines of code (LOC): 87

# lines of comment: 1

Percentage Comment: 1,136364

Cyclomatic Complexity (CC): 81

# Methods: 64

# Fields: 8

Type Level: 6

Difficulty Level: 18

Implementation Time(in seconds): 5 386

Estimated Delivered Bugs: 0,70

Depth of inheritance: 0

# Children: 1

# Types used: 9

# Types using me: 2

Association Between Types (ABT): 19

Type Rank: 0.33

Lack of Cohesion Of Methods (LCOM): 0.85

LCOM Henderson-Sellers (LCOMHS): 0.86

## Class Info

public class C\_engineer : C\_sl\_date, C\_sl\_personalys,  
C\_sl\_relations

- Drzewo\_genealogiczne

# Issues (Cumulated): 4

Debt Rating: **A**

All Debt: 13min

All Annual Interest: 12min

Breaking Point: 12 months

# lines of code (LOC): 117

# lines of comment: 8

Percentage Comment: 6,4

Cyclomatic Complexity (CC): 55

# Methods: 17

# Fields: 2

Type Level: 8

Difficulty Level: 39

Implementation Time(in seconds): 16 605

Estimated Delivered Bugs: 1,49

Depth of inheritance: 2

# Children: 2

# Types used: 29

# Types using me: 1

Association Between Types (ABT): 88

Type Rank: 0.31

Lack of Cohesion Of Methods (LCOM): 0.82

LCOM Henderson-Sellers (LCOMHS): 0.88

## Class Info

public class C\_first\_name : C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 2

Debt Rating: **A**

All Debt: 8min

All Annual Interest: 40min

Breaking Point: 2 months

# lines of code (LOC): 39

# lines of comment: 1

Percentage Comment: 2,5

Cyclomatic Complexity (CC): 27

# Methods: 14

# Fields: 1

Type Level: 3

Difficulty Level: 22

Implementation Time(in seconds): 2 087

Estimated Delivered Bugs: 0,37

Depth of inheritance: 1

# Children: 0

# Types used: 2

# Types using me: 4

Association Between Types (ABT): 13

Type Rank: 0.41

Lack of Cohesion Of Methods (LCOM): 0.29

LCOM Henderson-Sellers (LCOMHS): 0.31

## Class Info

public class C\_gender : C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 2

Debt Rating: **A**

All Debt: 8min

All Annual Interest: 40min

Breaking Point: 2 months

# lines of code (LOC): 32

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 30

# Methods: 13

# Fields: 1

Type Level: 3

Difficulty Level: 16

Implementation Time(in seconds): 1 242

Estimated Delivered Bugs: 0,26

Depth of inheritance: 1

# Children: 0

# Types used: 2

# Types using me: 4

Association Between Types (ABT): 9

Type Rank: 0.41

Lack of Cohesion Of Methods (LCOM): 0.31

LCOM Henderson-Sellers (LCOMHS): 0.33

## Class Info

public abstract class C\_goverment

- Drzewo\_genealogiczne

# Issues (Cumulated): 1

Debt Rating: **A**

All Debt: 0min 0s

All Annual Interest: 0min 0s

Breaking Point: 0 milli-second

# lines of code (LOC): 14

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 15

# Methods: 10

# Fields: 1

Type Level: 2

Difficulty Level: 10

Implementation Time(in seconds): 210

Estimated Delivered Bugs: 0,08

Depth of inheritance: 0

# Children: 3

# Types used: 1

# Types using me: 3

Association Between Types (ABT): 3

Type Rank: 0.4

Lack of Cohesion Of Methods (LCOM): 0.7

LCOM Henderson-Sellers (LCOMHS): 0.78

## Class Info

public class C\_government\_date : C\_government

- Drzewo\_genealogiczne

# Issues (Cumulated): 2

Debt Rating: **A**

All Debt: 4min 0s

All Annual Interest: 20min

Breaking Point: 2 months

# lines of code (LOC): 168

# lines of comment: 28

Percentage Comment: 14,28571

Cyclomatic Complexity (CC): 77

# Methods: 19

# Fields: 2

Type Level: 5

Difficulty Level: 84

Implementation Time(in seconds): 41 234

Estimated Delivered Bugs: 2,73

Depth of inheritance: 1

# Children: 0

# Types used: 7

# Types using me: 2

Association Between Types (ABT): 42

Type Rank: 0.23

Lack of Cohesion Of Methods (LCOM): 0.11

LCOM Henderson-Sellers (LCOMHS): 0.11

## Class Info

public class C\_government\_personaly : C\_government

- Drzewo\_genealogiczne

# Issues (Cumulated): 3

Debt Rating: **A**

All Debt: 6min

All Annual Interest: 20min

Breaking Point: 3 months

# lines of code (LOC): 108

# lines of comment: 24

Percentage Comment: 18,18182

Cyclomatic Complexity (CC): 56

# Methods: 15

# Fields: 2

Type Level: 4

Difficulty Level: 75

Implementation Time(in seconds): 23 022

Estimated Delivered Bugs: 1,85

Depth of inheritance: 1

# Children: 0

# Types used: 7

# Types using me: 2

Association Between Types (ABT): 36

Type Rank: 0.24

Lack of Cohesion Of Methods (LCOM): 0.17

LCOM Henderson-Sellers (LCOMHS): 0.18

## Class Info

public class **C\_government\_relation** : **C\_government**

- Drzewo\_genealogiczne

# Issues (Cumulated): 10

Debt Rating: **A**

All Debt: 1h 22min

All Annual Interest: 1h 33min

Breaking Point: 10 months

# lines of code (LOC): 214

# lines of comment: 34

Percentage Comment: 13,70968

Cyclomatic Complexity (CC): 88

# Methods: 19

# Fields: 2

Type Level: 6

Difficulty Level: 98

Implementation Time(in seconds): 56 600

Estimated Delivered Bugs: 3,37

Depth of inheritance: 1

# Children: 0

# Types used: 11

# Types using me: 2

Association Between Types (ABT): 55

Type Rank: 0.22

Lack of Cohesion Of Methods (LCOM): -0.18

LCOM Henderson-Sellers (LCOMHS): -0.19

## Class Info

public class **C\_grandchildren** : **C\_relation**

- Drzewo\_genealogiczne

# Issues (Cumulated): 6

Debt Rating: **A**

All Debt: 12min

All Annual Interest: 1h 56min

Breaking Point: 38 days

# lines of code (LOC): 48

# lines of comment: 1

Percentage Comment: 2,040816

Cyclomatic Complexity (CC): 31

# Methods: 16

# Fields: 3

Type Level: 5

Difficulty Level: 33

Implementation Time(in seconds): 3 703

Estimated Delivered Bugs: 0,55

Depth of inheritance: 1

# Children: 0

# Types used: 3

# Types using me: 4

Association Between Types (ABT): 20

Type Rank: 0.36

Lack of Cohesion Of Methods (LCOM): 0.54

LCOM Henderson-Sellers (LCOMHS): 0.58

## Class Info

public class C\_grandparents : C\_relation

- Drzewo\_genealogiczne

# Issues (Cumulated): 6

Debt Rating: **A**

All Debt: 12min

All Annual Interest: 1h 56min

Breaking Point: 38 days

# lines of code (LOC): 48

# lines of comment: 1

Percentage Comment: 2,040816

Cyclomatic Complexity (CC): 31

# Methods: 16

# Fields: 3

Type Level: 5

Difficulty Level: 33

Implementation Time(in seconds): 3 703

Estimated Delivered Bugs: 0,55

Depth of inheritance: 1

# Children: 0

# Types used: 3

# Types using me: 4

Association Between Types (ABT): 20

Type Rank: 0.36

Lack of Cohesion Of Methods (LCOM): 0.54

LCOM Henderson-Sellers (LCOMHS): 0.58

## Class Info

public class C\_human

- Drzewo\_genealogiczne

# Issues (Cumulated): 16

Debt Rating: **D**

All Debt: 1d 3h

All Annual Interest: 2h 43min

Breaking Point: 4 years

# lines of code (LOC): 292

# lines of comment: 1

Percentage Comment: 0,3412969

Cyclomatic Complexity (CC): 150

# Methods: 45

# Fields: 5

Type Level: 5

Difficulty Level: 120

Implementation Time(in seconds): 171 896

Estimated Delivered Bugs: 7,08

Depth of inheritance: 0

# Children: 0

# Types used: 11

# Types using me: 3

Association Between Types (ABT): 67

Type Rank: 0.41

Lack of Cohesion Of Methods (LCOM): 0.79

LCOM Henderson-Sellers (LCOMHS): 0.8



## Class Info

public class C\_id : C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 2

Debt Rating: **A**

All Debt: 8min

All Annual Interest: 40min

Breaking Point: 2 months

# lines of code (LOC): 38

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 27

# Methods: 17

# Fields: 1

Type Level: 3

Difficulty Level: 30

Implementation Time(in seconds): 2 778

Estimated Delivered Bugs: 0,45

Depth of inheritance: 1

# Children: 0

# Types used: 2

# Types using me: 15

Association Between Types (ABT): 12

Type Rank: 3.53

Lack of Cohesion Of Methods (LCOM): 0.29

LCOM Henderson-Sellers (LCOMHS): 0.31

## Class Info

public class C\_last\_name : C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 2

Debt Rating: **A**

All Debt: 8min

All Annual Interest: 40min

Breaking Point: 2 months

# lines of code (LOC): 39

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 27

# Methods: 14

# Fields: 1

Type Level: 3

Difficulty Level: 22

Implementation Time(in seconds): 2 087

Estimated Delivered Bugs: 0,37

Depth of inheritance: 1

# Children: 0

# Types used: 2

# Types using me: 4

Association Between Types (ABT): 13

Type Rank: 0.41

Lack of Cohesion Of Methods (LCOM): 0.29

LCOM Henderson-Sellers (LCOMHS): 0.31

## Class Info

public class **C\_month** : **C\_data**

- Drzewo\_genealogiczne

[# Issues \(Cumulated\)](#): 1

[Debt Rating](#): **A**

[All Debt](#): 6min

[All Annual Interest](#): 20min

[Breaking Point](#): 3 months

[# lines of code \(LOC\)](#): 36

[# lines of comment](#): 0

[Percentage Comment](#): 0%

[Cyclomatic Complexity \(CC\)](#): 26

[# Methods](#): 17

[# Fields](#): 1

[Type Level](#): 3

[Difficulty Level](#): 20

[Implementation Time\(in seconds\)](#): 1 686

[Estimated Delivered Bugs](#): 0,32

[Depth of inheritance](#): 1

[# Children](#): 1

[# Types used](#): 2

[# Types using me](#): 4

[Association Between Types \(ABT\)](#): 11

[Type Rank](#): 0.58

[Lack of Cohesion Of Methods \(LCOM\)](#): -0.12

[LCOM Henderson-Sellers \(LCOMHS\)](#): -0.12

## Class Info

public class **C\_order** : **C\_relation**

- Drzewo\_genealogiczne

[# Issues \(Cumulated\)](#): 5

[Debt Rating](#): **A**

[All Debt](#): 10min

[All Annual Interest](#): 1h 14min

[Breaking Point](#): 49 days

[# lines of code \(LOC\)](#): 51

[# lines of comment](#): 17

[Percentage Comment](#): 25%

[Cyclomatic Complexity \(CC\)](#): 35

[# Methods](#): 18

[# Fields](#): 4

[Type Level](#): 5

[Difficulty Level](#): 33

[Implementation Time\(in seconds\)](#): 4 180

[Estimated Delivered Bugs](#): 0,59

[Depth of inheritance](#): 1

[# Children](#): 0

[# Types used](#): 3

[# Types using me](#): 4

[Association Between Types \(ABT\)](#): 23

[Type Rank](#): 0.36

[Lack of Cohesion Of Methods \(LCOM\)](#): 0.64

[LCOM Henderson-Sellers \(LCOMHS\)](#): 0.68

## Class Info

public class C\_parent : C\_relation

- Drzewo\_genealogiczne

# Issues (Cumulated): 6

Debt Rating: **A**

All Debt: 12min

All Annual Interest: 1h 56min

Breaking Point: 38 days

# lines of code (LOC): 48

# lines of comment: 1

Percentage Comment: 2,040816

Cyclomatic Complexity (CC): 31

# Methods: 16

# Fields: 3

Type Level: 5

Difficulty Level: 33

Implementation Time(in seconds): 3 703

Estimated Delivered Bugs: 0,55

Depth of inheritance: 1

# Children: 0

# Types used: 3

# Types using me: 4

Association Between Types (ABT): 20

Type Rank: 0.36

Lack of Cohesion Of Methods (LCOM): 0.54

LCOM Henderson-Sellers (LCOMHS): 0.58

## Class Info

public class C\_partner : C\_relation

- Drzewo\_genealogiczne

# Issues (Cumulated): 6

Debt Rating: **A**

All Debt: 12min

All Annual Interest: 1h 56min

Breaking Point: 38 days

# lines of code (LOC): 48

# lines of comment: 1

Percentage Comment: 2,040816

Cyclomatic Complexity (CC): 31

# Methods: 16

# Fields: 3

Type Level: 5

Difficulty Level: 33

Implementation Time(in seconds): 3 703

Estimated Delivered Bugs: 0,55

Depth of inheritance: 1

# Children: 0

# Types used: 3

# Types using me: 4

Association Between Types (ABT): 20

Type Rank: 0.36

Lack of Cohesion Of Methods (LCOM): 0.54

LCOM Henderson-Sellers (LCOMHS): 0.58

## Class Info

public abstract class C\_relation

- Drzewo\_genealogiczne

# Issues (Cumulated): 3

Debt Rating: **B**

All Debt: 22min

All Annual Interest: 22min

Breaking Point: 12 months

# lines of code (LOC): 16

# lines of comment: 1

Percentage Comment: 5,882353

Cyclomatic Complexity (CC): 18

# Methods: 13

# Fields: 1

Type Level: 4

Difficulty Level: 10

Implementation Time(in seconds): 254

Estimated Delivered Bugs: 0,09

Depth of inheritance: 0

# Children: 7

# Types used: 2

# Types using me: 8

Association Between Types (ABT): 4

Type Rank: 2.01

Lack of Cohesion Of Methods (LCOM): 0.62

LCOM Henderson-Sellers (LCOMHS): 0.67

## Class Info

public abstract class C\_save\_load

- Drzewo\_genealogiczne

# Issues (Cumulated): 1

Debt Rating: **A**

All Debt: 0min 0s

All Annual Interest: 0min 0s

Breaking Point: 0 milli-second

# lines of code (LOC): 13

# lines of comment: 1

Percentage Comment: 7,142857

Cyclomatic Complexity (CC): 13

# Methods: 8

# Fields: 0

Type Level: 2

Difficulty Level: 5

Implementation Time(in seconds): 68

Estimated Delivered Bugs: 0,04

Depth of inheritance: 0

# Children: 12

# Types used: 1

# Types using me: 3

Association Between Types (ABT): 0

Type Rank: 0.32

Lack of Cohesion Of Methods (LCOM): 0

LCOM Henderson-Sellers (LCOMHS): 0

## Class Info

public class C\_sibling : C\_relation

- Drzewo\_genealogiczne

# Issues (Cumulated): 5

Debt Rating: **A**

All Debt: 10min

All Annual Interest: 1h 14min

Breaking Point: 49 days

# lines of code (LOC): 47

# lines of comment: 15

Percentage Comment: 24,19355

Cyclomatic Complexity (CC): 31

# Methods: 16

# Fields: 3

Type Level: 5

Difficulty Level: 33

Implementation Time(in seconds): 3 627

Estimated Delivered Bugs: 0,54

Depth of inheritance: 1

# Children: 0

# Types used: 3

# Types using me: 4

Association Between Types (ABT): 20

Type Rank: 0.36

Lack of Cohesion Of Methods (LCOM): 0.56

LCOM Henderson-Sellers (LCOMHS): 0.6

## Class Info

public class C\_sl\_date : C\_save\_load

- Drzewo\_genealogiczne

# Issues (Cumulated): 5

Debt Rating: **B**

All Debt: 44min

All Annual Interest: 13min

Breaking Point: 3 years

# lines of code (LOC): 88

# lines of comment: 22

Percentage Comment: 20%

Cyclomatic Complexity (CC): 33

# Methods: 13

# Fields: 1

Type Level: 6

Difficulty Level: 40

Implementation Time(in seconds): 8 664

Estimated Delivered Bugs: 0,97

Depth of inheritance: 1

# Children: 3

# Types used: 14

# Types using me: 3

Association Between Types (ABT): 51

Type Rank: 0.4

Lack of Cohesion Of Methods (LCOM): 0.23

LCOM Henderson-Sellers (LCOMHS): 0.25

## Class Info

public class C\_sl\_personalys : C\_save\_load

- Drzewo\_genealogiczne

# Issues (Cumulated): 4

Debt Rating: **A**

All Debt: 7min

All Annual Interest: 12min

Breaking Point: 7 months

# lines of code (LOC): 83

# lines of comment: 19

Percentage Comment: 18,62745

Cyclomatic Complexity (CC): 31

# Methods: 12

# Fields: 1

Type Level: 5

Difficulty Level: 36

Implementation Time(in seconds): 7 041

Estimated Delivered Bugs: 0,84

Depth of inheritance: 1

# Children: 3

# Types used: 13

# Types using me: 3

Association Between Types (ABT): 47

Type Rank: 0.4

Lack of Cohesion Of Methods (LCOM): 0.25

LCOM Henderson-Sellers (LCOMHS): 0.27

## Class Info

public class C\_sl\_relations : C\_save\_load

- Drzewo\_genealogiczne

# Issues (Cumulated): 6

Debt Rating: **C**

All Debt: 2h 9min

All Annual Interest: 29min

Breaking Point: 4 years

# lines of code (LOC): 117

# lines of comment: 16

Percentage Comment: 12,03008

Cyclomatic Complexity (CC): 39

# Methods: 12

# Fields: 1

Type Level: 7

Difficulty Level: 45

Implementation Time(in seconds): 14 249

Estimated Delivered Bugs: 1,35

Depth of inheritance: 1

# Children: 3

# Types used: 18

# Types using me: 3

Association Between Types (ABT): 70

Type Rank: 0.4

Lack of Cohesion Of Methods (LCOM): 0.083

LCOM Henderson-Sellers (LCOMHS): 0.091

## Class Info

public class **N\_striing**

- Drzewo\_genealogiczne

# Issues (Cumulated): 20

Debt Rating: **B**

All Debt: 4h 40min

All Annual Interest: 1h 36min

Breaking Point: 35 months

# lines of code (LOC): 408

# lines of comment: 19

Percentage Comment: 4,449649

Cyclomatic Complexity (CC): 182

# Methods: 44

# Fields: 2

Type Level: 1

Difficulty Level: 105

Implementation Time(in seconds): 120 819

Estimated Delivered Bugs: 5,59

Depth of inheritance: 0

# Children: 0

# Types used: 2

# Types using me: 28

Association Between Types (ABT): 5

Type Rank: 14.26

Lack of Cohesion Of Methods (LCOM): -1.01

LCOM Henderson-Sellers (LCOMHS): -1.03

## Class Info

public class **C\_tree** : C\_element

- Drzewo\_genealogiczne

# Issues (Cumulated): 10

Debt Rating: **B**

All Debt: 2h 54min

All Annual Interest: 2h 32min

Breaking Point: 13 months

# lines of code (LOC): 212

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 88

# Methods: 16

# Fields: 8

Type Level: 7

Difficulty Level: 99

Implementation Time(in seconds): 37 998

Estimated Delivered Bugs: 2,59

Depth of inheritance: 1

# Children: 0

# Types used: 4

# Types using me: 1

Association Between Types (ABT): 20

Type Rank: 0.18

Lack of Cohesion Of Methods (LCOM): 0.36

LCOM Henderson-Sellers (LCOMHS): 0.38

## Class Info

public class N\_vektor<T>

- Drzewo\_genealogiczne

# Issues (Cumulated): 3

Debt Rating: **A**

All Debt: 2min 0s

All Annual Interest: 2min 0s

Breaking Point: 12 months

# lines of code (LOC): 116

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 49

# Methods: 16

# Fields: 2

Type Level: 0

Difficulty Level: 60

Implementation Time(in seconds): 14 965

Estimated Delivered Bugs: 1,39

Depth of inheritance: 0

# Children: 0

# Types used: 0

# Types using me: 11

Association Between Types (ABT): 0

Type Rank: 1.5

Lack of Cohesion Of Methods (LCOM): -0.81

LCOM Henderson-Sellers (LCOMHS): -0.87

## Class Info

public class C\_year : C\_data

- Drzewo\_genealogiczne

# Issues (Cumulated): 1

Debt Rating: **A**

All Debt: 6min

All Annual Interest: 20min

Breaking Point: 3 months

# lines of code (LOC): 34

# lines of comment: 0

Percentage Comment: 0%

Cyclomatic Complexity (CC): 25

# Methods: 17

# Fields: 1

Type Level: 3

Difficulty Level: 20

Implementation Time(in seconds): 1 510

Estimated Delivered Bugs: 0,30

Depth of inheritance: 1

# Children: 1

# Types used: 2

# Types using me: 4

Association Between Types (ABT): 11

Type Rank: 0.58

Lack of Cohesion Of Methods (LCOM): -0.12

LCOM Henderson-Sellers (LCOMHS): -0.12



### 5.3. Kod składający się z przetestowanych modułów:

#### *date.h*

```

/*****
*****
*"Date.h"
*
*
*
*
*CONTENTS:
* "Klasa zawierająca klasy: C_day, C_month, C_year"
*HISTORY:
*version    Date Changes
*
*1.0        30.04.2017  Original design
*
*1.1        02.05.2015   Adding a virtual destructor
*
*1.2        12.05.2015   Adding methods
*
*****
*****/
#ifndef C_DATE_H
#define C_DATE_H
#include "day.h"
#include "month.h"
#include "year.h"
class C_date : virtual public C_day, virtual public C_month, virtual public C_year
{
public:
    C_date(); //konstruktor bezparametrowy
    C_date(char value); //konstruktor parametrowy
    C_date(const C_date &d); //konstruktor kopiujacy
    C_date& operator=(const C_date &d); //operator przypisania
    bool operator==(const C_date &d); //operator poronania ==
    bool operator!=(const C_date &d); //operator porownania !=
    C_day m_set_day(); //staw dzien do daty
    C_month m_set_month(); //wstaw miesiac do daty
    C_year m_set_year(); //wstaw rok do daty
    N_string m_set_DD_MM_YYYY(); //zwroc date w postaci stringa typu dzien -
miesiac - rok
    N_string m_set_MM_DD_YYYY(); //zwroc date w postaci stringa typu miesiac
- dzien - rok
    N_string m_set_YYYY_MM_DD(); //zwroc date w postaci stringa typu rok -
miesiac - dzien
    N_string m_set_YYYY_DD_MM(); //zwroc date w postaci stringa typu rok -
dzien - miesiac
    void m_get_DD_MM_YYYY(C_day& day, C_month& month, C_year & year); //wstaw
date z dniem miesiacem i rokiem
    void m_shift_day(C_day day); //zmien dzien
    void m_shift_day(int day); //zmien dzien

```

```

void m_shift_day(N_string day); //zmien dzien
void m_shift_month(C_month month); //zmien miesiac
void m_shift_month(int month); //zmien miesiac
void m_shift_month(N_string month); //zmien miesiac
void m_shift_year(C_year year); //zmien rok
void m_shift_year(int year); //zmien rok
void m_shift_year(N_string year); //zmien rok
void m_clear(); //wyczysc wszystkie dane daty
N_string m_what_type_date(); //zwroc typ daty np. malzenstwa, smierci czy
urodzenia
void m_shift_char(char value);
void m_get_type(N_string value); //wstaw znak podzialki
virtual ~C_date(); //destruktor wirtualny
friend std::ostream& operator<<(std::ostream &is, const C_date &d);
//przeciazenie operatora przesuniecia bitowego na wyjscie
private:
    char c_value; //zmienna typu char do podzialki
    N_string s_value; //zmienna typu string do typu daty
    void m_analitic_date(bool b_what); //analiza daty
};
#endif //!C_DATE

```

## *day.h*

```

/*****
*****
*"day.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes
*1.0        26.04.2017  Original design
*1.1        02.05.2015  shift name variable
*1.2        02.05.2015  Adding a virtual destructor
*1.3        02.05.2015  Adding a virtual methods
*1.4        02.05.2015  Adding parameter constructors
*1.5        12.05.2015  Redbuind m_is_there_contens
*1.6        13.05.2015  Adding a method "m_set_variable()"
*****
*****/

#ifndef DAY_H

```

```

#define DAY_H
#include "../Personaly/data.h"
class C_day: public C_data
{
public:
    C_day(); //konstruktor bezparametrowy
    C_day(N_striing &day); //kostruktor parametrowy przyjmujacy stringa
    C_day(int day); //konstruktor parametrowy przyjmujacy inta
    C_day(const C_day &C); //konstruktor kopiujacy
    C_day& operator=(const C_day &C); //operator przypisania
    bool operator==(const C_day &C); //operator porownania ==
    bool operator!=(const C_day &C); //operator porownania !=
    virtual ~C_day();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_striing &contens);
    virtual int m_set_variable();
    N_striing m_day_set();
    void m_get_day(N_striing &contens);
private:
    virtual N_striing m_is_there_contens(N_striing &Word);
    virtual N_striing m_set_contens();
protected:
    int m_set_value_day();
    void m_get_value_day(int value);
    int i_data_day = NULL;
};
#endif // !day_H

```

## ***month.h***

```

/*****
*****
*"month.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes
*1.0        26.04.2017  Original design
*1.1        02.05.2015  shift name variable
*1.2        02.05.2015  Adding a virtual destructor
*1.3        02.05.2015  Adding a virtual methods
*1.4        02.05.2015  Adding parameter constructors
*1.5        12.05.2015  Redbuind m_is_there_contens

```

			Author/Programmer
*1.0	26.04.2017	Original design	Mateusz Marchelewicz
*1.1	02.05.2015	shift name variable	Lukasz Witek vel Witkowski
*1.2	02.05.2015	Adding a virtual destructor	Lukasz Witek vel Witkowski
*1.3	02.05.2015	Adding a virtual methods	Lukasz Witek vel Witkowski
*1.4	02.05.2015	Adding parameter constructors	Lukasz Witek vel Witkowski
*1.5	12.05.2015	Redbuind m_is_there_contens	Lukasz Witek vel Witkowski

```

*1.6    13.05.2015    Adding a method "m_set_variable()"
                        Lukasz Witek vel Witkowski
*****
*****/

#ifndef MONTH_H
#define MONTH_H
#include "../Personalaly/data.h"
class C_month: public C_data
{
public:
    C_month();
    C_month(N_string &month);
    C_month(int month);
    C_month(const C_month &C);
    C_month& operator=(const C_month &C);
    bool operator==(const C_month &C);
    bool operator!=(const C_month &C);
    virtual ~C_month();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_string &contens);
    virtual int m_set_variable();
    N_string m_month_set();
    void m_get_month(N_string &contens);
private:
    virtual N_string m_is_there_contens(N_string &Word);
    virtual N_string m_set_contens();
protected:
    int m_set_value_month();
    void m_get_value_month(int value);
    int i_data_month;
};
#endif // !month_H

```

## *year.h*

```

/*****
*****
**"year.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes

```

*1.0	26.04.2017	Orginal design	Author/Programmer Mateusz Marchelewicz
*1.1	02.05.2015	shift name variable	Lukasz Witek vel Witkowski
*1.2	02.05.2015	Adding a virtual destructor	Lukasz Witek vel Witkowski

*1.3	02.05.2015	Adding a virtual methods	Lukasz Witek vel Witkowski
*1.4	02.05.2015	Adding parameter constructors	Lukasz Witek vel Witkowski
*1.5	12.05.2015	Redbuind m_is_there_contens	Lukasz Witek vel Witkowski
*1.6	13.05.2015	Adding a method "m_set_variable()"	Lukasz Witek vel Witkowski

```

*****
*****/
//

```

```

#ifndef YEAR_H
#define YEAR_H
#include "../Personaly/data.h"
class C_year: public C_data
{
public:
    C_year();
    C_year(N_string &year);
    C_year(int year);
    C_year(const C_year &C);
    C_year& operator=(const C_year &C);
    bool operator==(const C_year &C);
    bool operator!=(const C_year &C);
    virtual ~C_year();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_string &contens);
    virtual int m_set_variable();
    void m_get_year(N_string &contens);
    N_string m_year_set();
private:
    virtual N_string m_is_there_contens(N_string &Word);
    virtual N_string m_set_contens();
protected:
    int m_set_value_year();
    void m_get_value_year(int value);
    int i_data_year;
};
#endif // !year_H

```

## *data.h*

```

/*****
*****
*"data.h"
*
*
*
*
*
*CONTENTS:
* "Klasa bazowa"
*HISTORY:
*version    Date    Changes
                                     Author/Programmer
*1.0        22.04.2017  Original design                Lukasz Witek vel Witkowski
*1.1        02.05.2015  Adding a virtual destructor    Lukasz Witek vel Witkowski
*1.2        02.05.2015  Adding a virtual methods       Lukasz Witek vel Witkowski
*1.3        02.05.2015  Adding a method "m_what_type()" Lukasz Witek vel Witkowski
*1.4        13.05.2015  Adding a method "m_set_variable()"Lukasz Witek vel Witkowski

*****
*****/
#ifndef DATA_H
#define DATA_H
#include "../narzedzia/strring.h"
#include "../Helpful/definition.h"
class C_data
{
public:
    C_data(N_striing string);
    C_data(const C_data& data);
    C_data& operator=(const C_data& data);
    bool operator==(const C_data& data);
    bool operator!=(const C_data& data);
//    ~C_data();
    virtual ~C_data();
//Methods Virtuals
    virtual bool m_wchat_is()=0;
    virtual void m_get_contens(N_striing &contens) = 0;
    virtual N_striing m_set_contens() = 0;
    virtual int m_set_variable() = 0;
//Methods Implementate
    N_striing m_what_type();
private:
    virtual N_striing m_is_there_contens(N_striing &Word) = 0;
    N_striing s_data_text;
};
#endif // !DATA_H
//dodac metode virtualna zwracajaca wartosc z define.h zaczynajaca sie od 't' np.
t_first_name
//zwracac pod postacia inta

```

## *first\_name.h*

```

/*****
*****/
"first_name.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes

*1.0      22.04.2017  Original design                                Author/Programmer
                                                Lukasz Witek vel Witkowski

*1.1      02.05.2015   Adding a virtual destructor                Lukasz Witek vel Witkowski

*1.2      02.05.2015   Adding a virtual methods                  Lukasz Witek vel Witkowski

*1.3      09.05.2017   Adding a overloaded operators              Mateusz Marchelewicz

*1.4      13.05.2015   Adding a method "m_set_variable()"Lukasz Witek vel Witkowski

*****/
*****/
#ifndef C_FIRST_NAME_H
#define C_FIRST_NAME_H
#include "data.h"
class C_first_name :public C_data
{
public:
    C_first_name();
    C_first_name(N_string &first);
    C_first_name(const C_first_name &first);
    C_first_name& operator=(const C_first_name &first);
    bool operator==(const C_first_name &first);
    bool operator!=(const C_first_name &first);
    bool operator>(C_first_name &first);
    bool operator<(C_first_name &first);
    friend std::ostream& operator<<(std::ostream& is,const C_first_name
&first);
    //~C_first_name();
    virtual ~C_first_name();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_string &contens);
    virtual N_string m_set_contens();
    virtual int m_set_variable();
protected:
private:
    virtual N_string m_is_there_contens(N_string &word);
    N_string s_data_first_name;
};

```

```
#endif // !C_FIRST_NAME_H
```

### *gender.h*

```
/*
*****
*****
**"gender.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes
*
*1.0        26.04.2017  Original design                                Author/Programmer
                                                Mateusz Marchelewicz
*1.1        02.05.2015  Adding a virtual destructor                  Lukasz Witek vel Witkowski
*1.2        02.05.2015  Adding a virtual methods                    Lukasz Witek vel Witkowski
*1.3        02.05.2015  Adding parameter constructors              Lukasz Witek vel Witkowski
*1.4        13.05.2015  Adding a method "m_set_variable()"Lukasz Witek vel Witkowski
*****
*****/

#ifndef GENDER_H
#define GENDER_H
#include "data.h"
class C_gender: public C_data
{
public:
    C_gender();
    C_gender(bool gender);
    C_gender(N_string &gender);
    C_gender(const C_gender &C);
    C_gender& operator=(const C_gender &C);
    bool operator==(const C_gender &C);
    bool operator!=(const C_gender &C);
    friend std::ostream& operator<<(std::ostream& is, const C_gender &gender);
    virtual ~C_gender();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_string &contens);
    virtual N_string m_set_contens();
    virtual int m_set_variable();
protected:
private:
    virtual N_string m_is_there_contens(N_string &Word);
    N_string s_data_gender;
};
#endif // !GENDER_H
```



## *id.h*

```

/*****
*****
*"id.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes
*
*1.0      22.04.2017  Original design
*
*1.1      02.05.2015  Adding a virtual destructor
*
*1.2      02.05.2015  Adding a virtual methods
*
*1.3      02.05.2015  Adding parameter constructors
*
*1.3      09.05.2017  Adding a overloaded operators
*
*1.4      13.05.2015  Adding a method "m_set_variable()"
*
*1.5      14.05.2015  Update constructor class
*
*****
*****/
#ifndef ID_H
#define ID_H
#include "data.h"
class C_id:public C_data
{
public:
    C_id();
    C_id(char* id);
    C_id(N_string &id, bool t);
    C_id(int id);
    C_id(const C_id &C);
    C_id& operator=(const C_id &C);
    bool operator==(const C_id &C);
    bool operator!=(const C_id &C);
    bool operator>(C_id &id);
    bool operator<(C_id &id);
    virtual ~C_id();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_string &contens);
    virtual N_string m_set_contens();
    virtual int m_set_variable();
    void m_get_contens(int value);
private:
    virtual N_string m_is_there_contens(N_string &Word);

```

```

        N_string s_data_id;
};
#endif // !ID_H

```

## *last\_name.h*

```

/*****
*****
*"last_name.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po C_data"
*HISTORY:
*version    Date Changes
*
*1.0        25.04.2017  Original design                                Author/Programmer
                                                Mateusz Marchelewicz
*1.1        02.05.2015   Adding a virtual destructor                Lukasz Witek vel Witkowski
*1.2        02.05.2015   Adding a virtual methods                  Lukasz Witek vel Witkowski
*1.3        09.05.2017   Adding a overloaded operators              Mateusz Marchelewicz
*1.4        13.05.2015   Adding a method "m_set_variable()"Lukasz Witek vel Witkowski

*****
*****/
#ifndef C_LAST_NAME_H
#define C_LAST_NAME_H
#include "data.h"
class C_last_name : public C_data
{
public:
    C_last_name();
    C_last_name(N_string &last);
    C_last_name(const C_last_name &last);
    C_last_name& operator=(const C_last_name &last);
    bool operator==(const C_last_name &last);
    bool operator!=(const C_last_name &last);
    friend std::ostream& operator<<(std::ostream& is, C_last_name &last);
    bool operator<(C_last_name &last);
    bool operator>(C_last_name &last);
    virtual ~C_last_name();
    virtual bool m_wchat_is();
    virtual void m_get_contens(N_string &contens);
    virtual N_string m_set_contens();
    virtual int m_set_variable();
protected:
private:
    virtual N_string m_is_there_contens(N_string &word);
    N_string s_data_last_name;

```

```
};  
#endif // !C_LAST_NAME_H
```

*children.h*

```

/*****
*****
**"children.h"
**
**
**
**
**
**CONTENTS:
** "Klasa dziecko po klasie C_relation"
**HISTORY:
**version    Date    Changes
**
**1.0        30.04.2017  Original design
**
**1.1        02.05.2015  Adding a virtual destructor
**
**1.2        03.05.2015  Adding a virtual methods
**
**1.3        13.05.2015  Adding a method "m_set_variable()"
**
**1.4        17.05.2017  Adding priority methods
**
**1.5        26.05.2017  Adding private bool variable and method "m_set_bChild()"
**                        Lukas Janus
*****
*****/
#ifndef CHILDREN_H
#define CHILDREN_H
#include "relation.h"
class C_children :public C_relation
{
public:
    C_children();
    C_children(C_id &id);
    C_children(const C_id &id);
    C_children(const C_children &children);
    C_children& operator=(const C_children &children);
    bool operator==(const C_children &children);
    bool operator!=(const C_children &children);
    virtual void m_get_id(C_id &id);
    virtual C_id m_set_id();
    C_id m_set_index_id();
    virtual ~C_children();
    virtual int m_set_variable();
    virtual void m_get_complete_content(N_string data);
    void m_set_bChild(bool bChildm);
    virtual void m_get_complete_content(C_id index, C_id value);
    friend std::ostream& operator<<(std::ostream &is, const C_children &data);
    N_string m_get_content();

```

```

private:
    C_id ID_index;
    C_id ID_value;
    bool bChild;
};
#endif // !CHILDREN_H

```

## *grandchildren.h*

```

/*****
*****
*"grandchildren.h"
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_relation"
*HISTORY:
*version    Date Changes
*
*1.0        02.04.2017  Original design                Author/Programmer
*                                     Lukasz Witek vel Witkowski
*
*1.1        03.05.2015   Adding a virtual methods      Lukasz Witek vel Witkowski
*
*1.2        13.05.2015   Adding a method "m_set_variable()"Lukasz Witek vel Witkowski
*
*1.3        17.05.2017   Adding priority methods        Lukasz Witek vel Witkowski
*
*1.4        25.05.2017   Adding private bool variable and method "m_set_bGrandC()"
*                        Lukasz Janus
*****
*****/
#ifndef C_GRANDCHILDREN_H
#define C_GRANDCHILDREN_H
#include "relation.h"
class C_grandchildren :    public C_relation
{
public:
    C_grandchildren();
    C_grandchildren(C_id &id);
    C_grandchildren(const C_id &id);
    C_grandchildren(const C_grandchildren & grandchildren);
    C_grandchildren& operator=(const C_grandchildren& grandchildren);
    bool operator==(const C_grandchildren& grandchildren);
    bool operator!=(const C_grandchildren& grandchildren);
    virtual ~C_grandchildren();
    virtual void m_get_id(C_id &id);
    virtual C_id m_set_id();
    C_id m_set_index_id();
    virtual int m_set_variable();
    virtual void m_get_complete_content(N_string data);
    virtual void m_get_complete_content(C_id index, C_id value);

```

```

        friend std::ostream& operator<<(std::ostream &is, const C_grandchildren
&data);
        void m_set_bGrandC(bool bGrandCm);
        N_string m_get_content();
private:
        C_id ID_index;
        C_id ID_value;

        bool bGrandC;
};
#endif // !C_GRANDCHILDREN_H

```

### *grandparents.h*

```

/*****
*****
*"grandparents.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_relation"
*HISTORY:
*version    Date    Changes
*1.0        02.04.2017    Original design                Author/Programmer
                                Lukasz Witek vel Witkowski
*1.1        03.05.2015    Adding a virtual methods        Lukasz Witek vel Witkowski
*1.2        13.05.2015    Adding a method "m_set_variable()"Lukasz Witek vel Witkowski
*1.3        17.05.2017    Adding priority methods          Lukasz Witek vel Witkowski
*1.4        25.05.2017    Adding private bool variable and method "m_set_bGrandP()"
                                Lukasz Janus
*****
*****/
#ifndef C_GRANDPARENTS_H
#define C_GRANDPARENTS_H
#include "relation.h"
class C_grandparents :    public C_relation
{
public:
    C_grandparents();
    C_grandparents(C_id &id);
    C_grandparents(const C_id &id);
    C_grandparents(const C_grandparents & grandparents);
    C_grandparents& operator=(const C_grandparents& grandparents);
    bool operator==(const C_grandparents& grandparents);
    bool operator!=(const C_grandparents& grandparents);
    virtual ~C_grandparents();
    virtual void m_get_id(C_id &id);

```

```

        virtual C_id m_set_id();
        C_id m_set_index_id();
        virtual int m_set_variable();
        virtual void m_get_complete_content(N_striing data);
        virtual void m_get_complete_content(C_id index, C_id value);
        friend std::ostream& operator<<(std::ostream &is, const C_grandparents
&data);
        void m_set_bGrandP(bool bGrandPm);
        N_striing m_get_content();
private:
        C_id ID_index;
        C_id ID_value;

        bool bGrandP;
};
#endif // !C_GRANDPARENTS_H

```

## order.h

```

/*****
*****
*"order.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_relation"
*HISTORY:
*version    Date    Changes
*1.0        30.04.2017    Original design
*                                Author/Programmer
*                                Lukasz Witek vel Witkowski
*1.1        02.05.2015    Adding a virtual destructor
*                                Lukasz Witek vel Witkowski
*1.2        03.05.2015    Adding a virtual methods
*                                Lukasz Witek vel Witkowski
*1.3        13.05.2015    Adding a method "m_set_variable()"Lukasz Witek vel Witkowski
*1.4        17.05.2017    Adding priority methods
*                                Lukasz Witek vel Witkowski
*1.5        25.05.2017    Adding private bool variable and method "m_set_bSib()"
*                                Lukasz Janus
*****
*****/
#ifndef ORDER_H
#define ORDER_H
#include "relation.h"
class C_order :public C_relation
{
public:
    C_order(); //konstruktor bezparametrowy
    C_order(C_id &id); //konstruktor parametrowy

```

```

C_order(const C_id &id); //konstruktor parametrowy ??
C_order(const C_order &sib); //konstruktor kopiujacy
C_order& operator=(const C_order &sib); //operator przypisania
bool operator==(const C_order &sib); //operator porownania ==
bool operator!=(const C_order &sib); //operator porownania !=
virtual void m_get_id(C_id &id); //wstawia id do ID_value
virtual C_id m_set_id(); //zwraca ID_value
C_id m_set_index_id();
virtual int m_set_variable(); //zwraca wartosc w t_order
virtual ~C_order(); //wirtualny destruktor
virtual void m_get_complete_content(N_striing data); //analizuje i
podstawia wyluskane dane pod dane prywatne
virtual void m_get_complete_content(C_id index, C_id value); //wstawia
argumenty do danych prywatnych
void m_set_bSib(bool bSibm); //metoda dostępu do zmiennej prywatnej bSib
N_striing m_get_content(); // wysylanie zawartosci id
friend std::ostream& operator<<(std::ostream &is, const C_order &data);
//przesuniecie operatora przesuniecie bitowego na wyjscie
void m_get_atribut(N_striing atribut); //wstawianie atrybutu
N_striing m_get_atribut(); //zwraca wartosc atrybutu
private:
C_id ID_index; //Id humana wskaznikowego
C_id ID_value; //Id humana na drugim koncu relacji
bool bSib; //zmienna bool
N_striing satribut; //zmienna z wlasna nazwa relacji
};
#endif // !order_H

```

## *parent.h*

```
/******  
*****  
*"parent.h"  
*  
*  
*  
*  
*  
*CONTENTS:  
* "Klasa dziecko po klasie C_relation"  
*HISTORY:  
*version    Date Changes  
  
*1.0        30.04.2017  Original design                               Author/Programmer  
                                                Lukasz Witek vel Witkowski  
  
*1.1        02.05.2015   Adding a virtual destructor                Lukasz Witek vel Witkowski  
  
*1.2        03.05.2015   Adding a virtual methods                  Lukasz Witek vel Witkowski  
  
*1.3        13.05.2015   Adding a method "m_set_variable()"Lukasz Witek vel Witkowski  
  
*1.4        17.05.2017   Adding priority methods                    Lukasz Witek vel Witkowski  
  
*1.4        25.05.2017   Adding private bool variable and method "m_set_bParent()" Lukasz Janus  
*****  
*****/  
#ifndef PARENT_H  
#define PARENT_H  
#include "relation.h"  
#include "../Personaly/id.h"  
class C_parent :public C_relation  
{  
public:  
    C_parent();  
    C_parent(C_id &id);  
    C_parent(const C_id &id);  
    C_parent(const C_parent &parent);  
    C_parent& operator=(const C_parent &parent);  
    bool operator==(const C_parent &parent);  
    bool operator!=(const C_parent &parent);  
    virtual void m_get_id(C_id &id);  
    virtual C_id m_set_id();  
    C_id m_set_index_id();  
    virtual int m_set_variable();  
    virtual ~C_parent();  
    virtual void m_get_complete_content(N_striing data);  
    virtual void m_get_complete_content(C_id index, C_id value);  
    friend std::ostream& operator<<(std::ostream &is, const C_parent &data);  
    void m_set_bParent(bool bParentm);  
    N_striing m_get_content();  
private:  
    C_id ID_index;
```



```

        C_id ID_value;
        bool bParent;
};
#endif // !PARENT_H

```

## *partner.h*

```

/*****
*****
*"grandchildren.h"
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_relation"
*HISTORY:
*version    Date Changes
                                           Author/Programmer
*1.0        02.05.2017  Original design                Lukasz Witek vel Witkowski

*1.2        25.05.2017  Adding private bool variable and method "m_set_bPart()"
                    Lukasz Janus
*****
*****/
#ifndef C_partner_H
#define C_partner_H
#include "relation.h"
class C_partner :public C_relation
{
public:
    C_partner();
    C_partner(C_id &id);
    C_partner(const C_id &id);
    C_partner(const C_partner & partner);
    C_partner& operator=(const C_partner& partner);
    bool operator==(const C_partner& partner);
    bool operator!=(const C_partner& partner);
    virtual ~C_partner();
    void m_set_bPart(bool bPartm);
    virtual void m_get_id(C_id &id);
    virtual C_id m_set_id();
    C_id m_set_index_id();
    virtual int m_set_variable();
    virtual void m_get_complete_content(N_striing data);
    virtual void m_get_complete_content(C_id index, C_id value);
    friend std::ostream& operator<<(std::ostream &is, const C_partner &data);
    N_striing m_get_content();
private:
    C_id ID_index;
    C_id ID_value;
    bool bPart;
};
#endif // !C_partner_H

```

## *relation.h*

```

/*****
*****
*"relation.h"
*
*
*
*
*CONTENTS:
* "Klasa virtualno basowa"
*HISTORY:
*version    Date    Changes
*
*1.0        30.04.2017    Original design
*
*1.1        02.05.2015    Adding a virtual destructor
*
*1.2        03.05.2015    Adding a virtual methods
*
*****
*****/
#ifndef C_RELATION_H
#define C_RELATION_H
#include "..\narzedzia\striing.h"
#include "..\Personalny/id.h"
#include "..\Helpful/definition.h"
class C_relation
{
public:
    C_relation();
    C_relation(N_striing reall);
    C_relation(const C_relation &relat);
    C_relation& operator=(const C_relation &relat);
    bool operator==(const C_relation &relat);
    bool operator!=(const C_relation &relat);
    virtual void m_get_id(C_id &id) = 0;
    virtual C_id m_set_id() = 0;        //~C_relation();
    virtual ~C_relation();
    N_striing m_what_type();
    virtual int m_set_variable() = 0;
    virtual void m_get_complete_content(N_striing data) = 0;
    virtual void m_get_complete_content(C_id index, C_id value) = 0;
private:
    N_striing S_relation_text;
};
#endif // !C_RELATION_H

```

## *sibling.h*

```

/*****
*****/
"sibling.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_relation"
*HISTORY:
*version    Date Changes
                                     Author/Programmer
*1.0        30.04.2017  Original design                Lukasz Witek vel Witkowski
*1.1        02.05.2015  Adding a virtual destructor      Lukasz Witek vel Witkowski
*1.2        03.05.2015  Adding a virtual methods        Lukasz Witek vel Witkowski
*1.3        13.05.2015  Adding a method "m_set_variable()"Lukasz Witek vel Witkowski
*1.4        17.05.2017  Adding priority methods          Lukasz Witek vel Witkowski
*1.5        25.05.2017  Adding private bool variable and method "m_set_bSib()"
                        Lukasz Janus
*****/
*****/
#ifndef SIBLING_H
#define SIBLING_H
#include "relation.h"
class C_sibling:public C_relation
{
public:
    C_sibling(); //konstruktor bezparametrowy
    C_sibling(C_id &id); //konstruktor parametrowy
    C_sibling(const C_id &id); //konstruktor parametrowy ??
    C_sibling(const C_sibling &sib); //konstruktor kopiujacy
    C_sibling& operator=(const C_sibling &sib); //operator przypisania
    bool operator==(const C_sibling &sib); //operator porownania ==
    bool operator!=(const C_sibling &sib); //operator porownania !=
    virtual void m_get_id(C_id &id); //wstawia id do ID_value
    virtual C_id m_set_id(); //zwraca ID_value
    C_id m_set_index_id();
    virtual int m_set_variable(); //zwraca wartosc w t_sibling
    virtual ~C_sibling(); //wirtualny destruktor
    virtual void m_get_complete_content(N_striing data); //analizuje i
    podstawia wyluskane dane pod dane prywatne
    virtual void m_get_complete_content(C_id index, C_id value); //wstawia
    argumenty do danych prywatnych
    friend std::ostream& operator<<(std::ostream &is, const C_sibling &data);
    //przeciazenie operatora przesuniecia bitowego na wyjscie
    void m_set_bSib(bool bSibm); //metoda dostępu do zmiennej prywatnej bSib
    N_striing m_get_content();

```

```
private:
    C_id ID_index; //Id humana wskaznikowego
    C_id ID_value; //Id humana na drugim koncu relacji
    bool bSib;    //zmienna bool
};
#endif // !SIBLING_H
```

## *element.h*

```

/*****
*****
*"element.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_human zawiera tablice dynamiczna relacji"
*HISTORY:
*version    Date Changes
*
*1.0        01.05.2017  Original design
*
*1.1        02.05.2015  Adding a virtual destructor
*
*1.2        03.05.2015  adding methods
*
*****
*****/
#ifndef ELEMENT_H
#define ELEMENT_H
#include "human.h"
#include "../Relation/children.h"
#include "../Relation/sibling.h"
#include "../Relation/parent.h"
#include "../Relation/grandchildren.h"
#include "../Relation/grandparents.h"
#include "../Relation/partner.h"
#include "../Relation/order.h"
class C_element
{
public:
    C_element();
    C_element(const C_human& human);
    C_element(const C_element &human);
    C_element& operator=(const C_element &human);
    bool operator==(const C_element &human);
    bool operator!=(const C_element &human);
    void m_get_children(C_children &children);
    void m_get_parent(C_parent &parent);
    void m_get_sibling(C_sibling &sibling);
    void m_get_grandchildren(C_grandchildren &grandchildren);

```

```

void m_get_grandparents(C_grandparents &grandparents);
void m_get_partner(C_partner &partner);
void m_get_order(C_order &order);
void m_update_children(int value, C_children &children);
void m_update_parent(int value, C_parent &parent);
void m_update_sibling(int value, C_sibling &sibling);
void m_update_human(const C_human &human);
void m_update_grandchildren(int value, C_grandchildren &human);
void m_update_grandparents(int value, C_grandparents &human);
void m_update_partner(int value, C_partner &partner);
void m_update_order(int value, C_order &order);
C_human m_set_Human();
C_children m_set_children();
C_parent m_set_parent();
C_sibling m_set_sibling();
C_partner m_set_partner();
C_grandchildren m_set_grandchildren();
C_grandparents m_set_grandparents();
C_order m_set_order();
C_children m_set_children(int value);
C_parent m_set_parent(int value);
C_sibling m_set_sibling(int value);
C_grandchildren m_set_grandchildren(int value);
C_grandparents m_set_grandparents(int value);
C_partner m_set_partner(int value);
C_order m_set_order(int value);
C_element& m_clean();
void m_clean_children();
void m_clean_parent();
void m_clean_sibling();
void m_clean_grandparents();
void m_clean_grandchildren();
void m_clean_partner();
void m_clean_order();
void m_delete_children();
void m_delete_parent();
void m_delete_sibling();
void m_delete_partner();
void m_delete_order();
void m_delete_children(int value);
void m_delete_parent(int value);
void m_delete_sibling(int value);
void m_delete_grandchildren(int value);
void m_delete_grandparents(int value);
void m_delete_partner(int value);
void m_delete_order(int value);
N_vektor<C_grandparents> m_set_v_grandparents();
N_vektor<C_grandchildren> m_set_v_grandchildren();
N_vektor<C_parent> m_set_v_parent();
N_vektor<C_children> m_set_v_children();
N_vektor<C_partner> m_set_v_partner();
N_vektor<C_sibling> m_set_v_sibling();
N_vektor<C_order> m_set_v_order();
//~C_element();
virtual ~C_element();
protected:
    C_human Human;
private:

```

```

        N_vektor<C_children> V_children;
        N_vektor<C_parent> V_parent;
        N_vektor<C_sibling> V_sibling; //trzeba przetestowac czy dziala, jak nie to
cos wymysle!!
        N_vektor<C_grandchildren> V_grandchildren;
        N_vektor<C_grandparents> V_grandparents;
        N_vektor<C_partner> V_partner;
        N_vektor<C_order> V_order;
};
#endif // !ELEMENT_H

```

## ***government.h***

```

/*****
*****
*"government.h"
*
*
*
*
*
*CONTENTS:
* "Klasa bazowa dla innych klas do przechowywania 'wierszy' pliku"
*HISTORY:
*version    Date Changes
*
*1.0        06.05.2017  Original design
*
*1.01       09.05.2017  Pure Virtual Methods Added
*
*****
*****/
#ifndef GOVERNMENT_H
#define GOVERNMENT_H
#include "../narzedzia/string.h"
#include "../narzedzia/Vektor.h"
#include "../Helpful/definition.h"
class C_government
{
public:
    C_government();
    C_government(const C_government & government);
    C_government& operator=(const C_government& government);
    bool operator==(const C_government& government);
    bool operator!=(const C_government& government);
    virtual ~C_government();
    virtual bool m_wchat_is() = 0;
    virtual void m_get_contens(N_string &contens) = 0;
    virtual N_string m_set_contens() = 0;

private:
    virtual N_string m_is_there_contens(N_string &word) = 0;
    N_string s_government_text;
};
#endif // !GOVERNMENT_H

```

## *goverment\_date.h*

```

/*****
*****
*"goverment_date.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_goverment"
*HISTORY:
*version    Date Changes
*
*1.0        06.05.2017  Original design                                Author/Programmer
                                                Lukasz Witek vel Witkowski
*1.1        09.05.2017  Pure Virtual Methods Added                    Mateusz Marchelewicz
*1.2        12.05.2017  rebuild m_is_there_contens                    Lukasz Witek vel Witkowski
*1.3        12.05.2017  Adding new mothods                            Lukasz Witek vel Witkowski
*1.4        17.05.2017  Adding a method to return values from date classes
                        Lukasz Witek vel Witkowski
*****
*****/
#ifndef GOVERNMENT_DATE_H
#define GOVERNMENT_DATE_H
#include "goverment.h"
#include "../Date/date.h"
class C_goverment_date :
    public C_goverment
{
public:
    C_goverment_date(); //konstruktor bezparametrowy
    C_goverment_date(const C_goverment_date & goverment_date); //konstruktor
kopiujacy
    C_goverment_date& operator=(const C_goverment_date& goverment_date);
//operator przypisania =
    bool operator==(const C_goverment_date& goverment_date); //operator
porownania ==
    bool operator!=(const C_goverment_date& goverment_date); //operator
porownania !=
    virtual ~C_goverment_date(); //destruktor wirtualny
    virtual bool m_wchat_is(); //pokazuje czy istnieje jakis string
    virtual void m_get_contens(N_string &contens); //wprowadza string do
klasy
    virtual N_string m_set_contens(); //zwraca caly string
    int m_set_value_id(); //zwraca wyszukana wczesniej wartosc id
    C_day m_set_value_day(); //wyszukuje i zwraca wartosc day
    C_month m_set_value_month(); //wyszukuje i zwraca wartosc month
    C_year m_set_value_year(); //wyszukuje i zwraca wartosc year
    N_vektor<C_date> m_set_value_V_date(); // zwraca wektor obiektow C_date
    N_vektor<C_day> m_set_value_V_day(); //wyszukuje i zwraca wartosc day
    N_vektor<C_month> m_set_value_V_month(); //wyszukuje i zwraca wartosc month

```

```

        N_vektor<C_year> m_set_value_V_year(); //wyszukuje i zwraca wartosc year
private:
    virtual N_string m_is_there_contens(N_string &word); //analizuje
wprowadzany string pod wzgledem poprawnej zawartosci
    N_string s_goverment_data; //zmienna przechowujaca string
    int m_id_value(); //wyszukuje id
    int i_value_id; //zmienna przechowujaca id
};
#endif // !GOVERNMENT_DATE_H

```

### ***government\_personaly.h***

```

/*****
*****
*"government_personaly.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_government"
*HISTORY:
*version    Date    Changes
*1.0        06.05.2017    Original design
*1.1        09.05.2017    Pure Virtual Methods Added
*1.2        12.05.2017    redbuild m_is_there_contens
*1.3        12.05.2017    Adding new mothods
*1.4        17.05.2017    Adding a method to return values from personaly classes
                        Lukasz Witek vel Witkowski
*****
*****/
#ifndef GOVERNMENT_PERSONALY_H
#define GOVERNMENT_PERSONALY_H
#include "government.h"
#include "../Personaly/first_name.h"
#include "../Personaly/gender.h"
#include "../Personaly/last_name.h"
class C_government_personaly :
    public C_government
{
public:
    C_government_personaly(); //konstruktor beparametrowy
    C_government_personaly(const C_government_personaly & government_personaly);
//konstruktor kopiujacy
    C_government_personaly& operator=(const C_government_personaly&
government_personaly); //operator przypisania
    bool operator==(const C_government_personaly& government_personaly);
//operator porownania ==

```





```

#include "../Relation/grandparents.h"
#include "../Relation/parent.h"
#include "../Relation/sibling.h"
#include "../Relation/partner.h"
#include "../Relation/order.h"
class C_government_relation :
    public C_government
{
public:
    C_government_relation(); //konstruktor bezparametrowy
    C_government_relation(const C_government_relation & government_relation);
//konstruktor kopiujacy
    C_government_relation& operator=(const C_government_relation&
government_relation); //operator przypisania
    bool operator==(const C_government_relation& government_relation); //operator
porownania ==
    bool operator!=(const C_government_relation& government_relation); //operator
porownania !=
    virtual ~C_government_relation(); //wirtualny destruktork
    virtual bool m_what_is(); //sprawdza czy istnieje zawartosc stringa
    virtual void m_get_contens(N_string &contens); //wprowadza string
    virtual N_string m_set_contens(); //zwraca wartosc stringa
    int m_set_value_id(); //zwraca wyszukana wzczesniej wartosc stringa
    N_vektor<C_children> m_set_value_children(); //wyszukuje i zwraca wartosc
dla children
    N_vektor<C_parent> m_set_value_parent(); //wyszukuje i zwraca wartosc dla
parent
    N_vektor<C_grandchildren> m_set_value_grandchildren(); //wyszukuje i zwraca
wartosc dla grandchildren
    N_vektor<C_grandparents> m_set_value_grandparents(); //wyszukuje i zwraca
wartosc dla grandparent
    N_vektor<C_sibling> m_set_value_sibling(); //wyszukuje i zwraca wartosc dla
sibling
    N_vektor<C_partner> m_set_value_patner(); //wyszukuje i zwraca wartosc dla
sibling
    N_vektor<C_order> m_set_value_order();
private:
    virtual N_string m_is_there_contens(N_string &word); //analizuje
wprowadzany string
    N_string s_government_relation; //zmienna przechowujaca string
    int m_id_value(); //wyszukuje id
    int i_value_id; //przechowuje wartosc id
};
#endif // !GOVERNMENT_RELATION_H

//doxygen

```

## *human.h*

```

/*****
*****
*"human.h"
*
*
*
*
*CONTENTS:
* "Klasa reprezentujaca czlowieka, najmniejsza komurka bazy danych, zawierajaca:
C_first_name, C_last_name, C_id,
C_date"
*HISTORY:
*version    Date    Changes
*
*1.0        30.04.2017  Original design                                Author/Programmer
                                                                Lukasz Witek vel Witkowski
*1.1        02.05.2015   adding a virtual destructor                Lukasz Witek vel Witkowski
*1.2        02.05.2015   adding parameter constructor              Lukasz Witek vel Witkowski
*1.3        03.05.2015   adding methods                            Lukasz Witek vel Witkowski
*1.3        09.05.2017   Adding a overloaded operator              Mateusz Marchelewicz
*1.3        23.05.2017   Adding methods interface                  Bartosz Bukowski

*****
*****/
#ifndef C_HUMAN_H
#define C_HUMAN_H
#include "../Personalny/last_name.h"
#include "../Personalny/first_name.h"
#include "../Personalny/id.h"
#include "../date/date.h"
#include "../Personalny/gender.h"
#include "../narzedzia/Vektor.h"
#include <windows.h>
class C_human
{
public:
    C_human();
    C_human(C_id &id);
    C_human(const C_human &human);
    C_human& operator=(const C_human &human);
    bool operator==(const C_human &human);
    bool operator!=(const C_human &human);
    //friend std::ostream& operator<<(std::ostream& is, C_human &human);
    void m_get_first_name(C_first_name &f_name);
    void m_get_first_name(N_string &f_name);
    void m_get_last_name(C_last_name &l_name);
    void m_get_last_name(N_string &l_name);

```

```

void m_get_gender(C_gender &gender);
void m_get_gender(N_string &gender);
void m_get_gender(bool gender);
void m_shift_id(N_string &id);
void m_shift_id(int id);
void m_shift_id(C_id &id);
void m_get_date(C_date date);
void m_delete_first_name();
void m_delete_last_name(int value);
void m_delete_last_name();
void m_delete_gender();
void m_delete_date(int value);
void m_delete_date();
void m_update_date(int value, C_date& date);
void m_update_last_name(int value, C_last_name& l_name);
void m_update_last_name(int value, N_string& l_name);
void interf_cut(N_string &first, N_string &last, C_human &human, int
cut);
void interf_m(C_human &human, C_date &d, C_date ds = NULL);
void interf_mb(N_string firstamee, N_string lastname, C_date &du,
C_date ds = NULL, char poz = '*', char pion = '|');
void interf_mbd(N_string firstame, N_string lastname, C_date &du, C_date
ds = NULL, char poz = '*', char pion = '|');
friend std::ostream& operator<<(std::ostream &is, const C_human &h);
N_string m_short_interface_personaly();
N_string m_short_interface_date();
C_human& m_clear();
C_human& m_clear_date();
C_human& m_clear_last_name();
C_first_name m_set_first_name();
C_last_name m_set_last_name();
C_last_name m_set_last_name(int value);
C_gender m_set_gender();
C_id m_set_id();
C_date m_set_date(int value);
C_date m_set_date();
N_vektor<C_date> m_set_Vdate();
N_vektor<C_last_name> m_set_V_last_name();
virtual ~C_human();
private:
N_vektor<C_date> V_date;
C_id Id;
C_first_name First;
N_vektor<C_last_name> V_last;
C_gender Gender;
};
#endif // !C_human_H

```

## *tree.h*

```

/*****
*****
**"tree.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_element zawiera tablice dynamiczna ludzi"
*HISTORY:
*version    Date Changes
*
*1.0        01.05.2017  Original design                                Author/Programmer
                                                Lukasz Witek vel Witkowski
*1.1        02.05.2015   Adding a virtual destructor                Lukasz Witek vel Witkowski
*1.2        03.05.2015   adding methods                             Lukasz Witek vel Witkowski
*****
*****/
#ifndef TREE_H
#define TREE_H
#include "element.h"
class C_tree :public C_element
{
public:
    C_tree();
    C_tree(const C_element &element);
    C_tree(const C_human &human);
    C_tree(const C_tree &tree);
    C_tree& operator=(const C_tree &tree);
    bool operator==(const C_tree &tree);
    bool operator!=(const C_tree &tree);
    void m_add_human(C_human &h, int data, bool bwhat, int ivalue);
    void m_update_human(C_human &h, int data, int ivalue);
    void m_delete_human(C_human &h, int data, int ivalue);
    void m_delete_human(C_human &h, int data);
    void m_add_id(const C_id &id);
    C_id m_get_id();
    C_human m_get_human(int data, int ivalue);
    friend std::ostream& operator<<(std::ostream& is, const C_tree &tree);
    C_human m_get_index_human();
    virtual ~C_tree();
private:
    N_vektor<C_human> V_human_grandparent;
    N_vektor<C_human> V_human_grandchildren;
    N_vektor<C_human> V_human_children;
    N_vektor<C_human> V_human_parent;
    N_vektor<C_human> V_human_sibling;
    N_vektor<C_human> V_human_partner;
    N_vektor<C_human> V_human_order;
    C_id ID_tree;
};

```

```
#endif // !TREE_H
```

### *alphabet.h*

```
#ifndef ALPHABET_H
#define ALPHABET_H
#include "../narzedzia/Vektor.h"
struct alfabet
{
    char litera;
    int lp;
    int ascii;
    bool operator==(alfabet &a) {
        if (litera == a.litera&&lp == a.lp&&ascii == a.ascii) return true;
    }
    return false;
    bool operator!=(alfabet &a) {
        if (litera != a.litera&&lp != a.lp&&ascii != a.ascii) return true;
    }
    return false;
}
void ladowanie_sz(N_vektor<alfabet> &v)
{
    char lit = ' ';
    for (int i = 0; i < 222; i++)
    {
        alfabet t;
        t.litera = lit;
        t.lp = i + 1;
        t.ascii = (int)lit;
        v.m_push_back(t);
        lit++;
    }
};
#endif // !ALPHABET_H
```

## *enginer.h*

```
/*
*****
*****
*/
"enginer.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_tree reprezentuje silnik aplikacji"
*HISTORY:
*version    Date Changes
*
*1.0        01.05.2017  Original design
*
*1.1        02.05.2015  Adding a virtual destructor
*
*****
*****/
#ifndef ENGINER_H
#define ENGINER_H
#include "../Databases/tree.h"
#include "sl_date.h"
#include "sl_personalys.h"
#include "sl_relations.h"
class C_enginer :virtual public C_sl_date, virtual public C_sl_personalys, virtual
public C_sl_relations
{
public:
    C_enginer();
    C_enginer(const C_enginer &enginer);
    C_enginer& operator=(const C_enginer &enginer);
    bool operator==(const C_enginer &enginer);
    bool operator!=(const C_enginer &enginer);
    virtual ~C_enginer();
    int m_set_index();
    void m_load_files();
    void m_save_files();
    void m_new_human(C_human &human);
    void m_new_element(C_element &element, bool b_what);
    C_element m_create_element(C_id id_finter);
    C_human m_create_human(C_id id_finter);
    C_tree m_create_tree(C_id id_pointer);
private:
    void m_printer(int i);
    void m_get_index(int value);
    void m_file_init(bool b_what);
    N_vektor<C_tree> V_tree;
    static int i_index;
};
#endif // !ENGINER_H
```

## *save\_load.h*

```

/*****
*****
*"save_load.h"
*
*
*
*
*
*CONTENTS:
* "Klasa bazowa dla innych klas do wczytywania"
*HISTORY:
*version    Date Changes
*1.0        06.05.2017  Original design
                                           Author/Programmer
                                           Lukasz Witek vel Witkowski

*****
*****/
#ifndef C_SAVE_LOAD_H
#define C_SAVE_LOAD_H
#include "alphabet.h"
#include "../narzedzia/string.h"
#include "../narzedzia/Vektor.h"
#include "../Helpful/definition.h"
#include <fstream>
class C_save_load
{
public:
    C_save_load();
    C_save_load(const C_save_load & save_load);
    C_save_load& operator=(const C_save_load& save_load);
    bool operator==(const C_save_load& save_load);
    bool operator!=(const C_save_load& save_load);
    virtual ~C_save_load();
    virtual N_string m_cypher_on(N_string data)=0; //odszyfrowywanie
    virtual N_string m_cypher_off(N_string data)=0; //zaszyfrowywanie
    /*void wczytuj_sz(N_string& slowo, N_vektor<N_string>& s);
    void zapis_sz(N_string a, N_string& haslo, int* s);
    N_string m_encryption_sz(N_vektor<alfabet>& v, N_string& haslo,
N_vektor<N_string>& s);
    int main_m_encryption(N_string a, N_string tablica, int X);
    void m_analyzing(N_vektor <N_string> &date, N_string files);
    int main_rozkodowywanie(N_string a);
    void wczytuj(N_string& haslo, bool& istnieje);
    void zapis(N_string a, N_string& haslo);
    void wczytaj(N_string a, N_string& slowo);
    N_string m_encryption(N_string& slowo, N_vektor<alfabet>& v, N_string&
haslo);
    void wczytaj_rozkodowacz(N_string a, N_vektor <N_string> &date, bool&
istnieje);
    N_string m_decryption(N_string a, N_string tablica, int X);*/
};
#endif // !C_SAVE_LOAD_H
```



## *sl\_date.h*

```

/*****
*****
*"sl_date.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_save_load"
*HISTORY:
*version    Date Changes
*1.0        06.05.2017  Original design
                                           Author/Programmer
                                           Lukasz Witek vel Witkowski
*****
*****/
#ifndef C_SL_DATE_H
#define C_SL_DATE_H
#include "save_load.h"
#include "../narzedzia/Vektor.h"
#include "../Databases/government_date.h"
#include "../Personal/id.h"
class C_sl_date :public C_save_load
{
public:
    C_sl_date(); //konstruktor bezparametrowy
    C_sl_date(const C_sl_date & sl_date); //konstruktor kopiujacy
    C_sl_date& operator=(const C_sl_date& sl_date); //operator przypisania
    bool operator==(const C_sl_date& sl_date); //operator porownania ==
    bool operator!=(const C_sl_date& sl_date); //operator porownania !=
    virtual ~C_sl_date(); //wirtualny destruktor
protected:
    void m_file_date(bool what); //czytanie z pliku
    bool m_what(int value);
    virtual N_string m_cypher_on(N_string data); //odszyfrowywanie
    virtual N_string m_cypher_off(N_string data); //zaszyfrowywanie
    void m_get_new_date(C_id id, N_vektor<C_date> V_date); //dodawanie nowych
    dat
    C_government_date& operator[](int i);
    C_government_date m_set_gover_date(int i);
    N_vektor<C_government_date> V_government_date;
};
#endif // !C_SL_DATE_H
```

*sl\_personalys.h*

```

/*****
*****
**sl_date.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_save_load"
**HISTORY:
*version    Date Changes
                                           Author/Programmer
*1.0        06.05.2017   Original design                Lukasz Witek vel Witkowski

*****/
#ifndef C_SL_PERSONALYS_H
#define C_SL_PERSONALYS_H
#include "save_load.h"
#include "../Personalys/id.h"
#include "../Databases/goverment_personalys.h"
#include "../narzedzia/Vektor.h"
class C_sl_personalys :
public C_save_load
{
public:
    C_sl_personalys(); //konstruktor bezparametrowy
    C_sl_personalys(const C_sl_personalys & sl_personalys); // konstruktor
kopiujacy
    C_sl_personalys& operator=(const C_sl_personalys& sl_personalys);
//operator przypisania
    bool operator==(const C_sl_personalys& sl_personalys); //operator
porownania ==
    bool operator!=(const C_sl_personalys& sl_personalys); //operator
porownania !=
    virtual ~C_sl_personalys(); //destruktor virtualny
protected:
    bool m_what(int value);
    void m_load_file_personalys(bool what); //czytanie z pliku
    virtual N_string m_cypher_on(N_string data); //odszyfrowywanie
    virtual N_string m_cypher_off(N_string data); //zaszyfrowywanie
    void m_add_new_personalys(C_id id, C_first_name first, N_vektor<C_last_name>
Last, C_gender gender); //dodawanie nowych danych personalnych
    C_goverment_personalys& operator[](int i); //operator "dostepu" []
    N_vektor<C_goverment_personalys> V_goverment_personalys;
};
#endif // !C SL PERSOLALYS H

```

```
/*****  
*****  
*****  
*****/  
*"sl_date.h"  
*  
*  
*  
*  
*  
*CONTENTS:  
* "Klasa dziecko po klasie C_save_load"  
*HISTORY:  
*version    Date Changes  
  
*1.0        06.05.2017   Original design  
  
Author/Programmer  
Lukasz Witek vel Witkowski  
  
*****  
*****/  
#ifndef C_SL_RELATIONS  
#define C_SL_RELATIONS  
#include "save_load.h"  
#include "../Databases/goverment_relation.h"  
#include "../Relation/partner.h"  
#include "../narzedzia/Vektor.h"  
class C_sl_relations :public C_save_load  
{  
public:  
    C_sl_relations(); //konstruktor bezparametrowy  
    C_sl_relations(const C_sl_relations & sl_relations); //konstruktor  
kopiujacy  
    C_sl_relations& operator=(const C_sl_relations& sl_relations); //operator  
przypisania  
    bool operator==(const C_sl_relations& sl_relations); //operator porownania  
==  
    bool operator!=(const C_sl_relations& sl_relations); //operator porownania  
!=  
    virtual ~C_sl_relations(); //wirtualny destruktor  
protected:  
    void m_load_file_relation(bool what); //czytanie z pliku  
    virtual N_string m_cypher_on(N_string data); //odszyfrowywanie  
    virtual N_string m_cypher_off(N_string data); //zaszyfrowywanie  
    void m_add_new_relations(C_id id,N_vektor<C_children> V_children,  
N_vektor<C_parent> V_parent, N_vektor<C_sibling> V_sibling,  
N_vektor<C_grandchildren> V_grandchildren, N_vektor<C_grandparents>  
V_grandparents,  
N_vektor<C_partner> V_partners, N_vektor<C_order> V_order);  
    N_vektor<C_government_relation> V_government_relation;  
    C_government_relation m_set_gover_relation(int i);  
private:  
    bool m_what(int i);  
};  
#endif // !C_SL_RELATIONS
```

***definition.h***

```

/******
*****
*/
#define definition.h
*
*
*
*
*
*
*CONTENTS:
* "Definicje header'ów klas"
*HISTORY:
*version    Date Changes                                     Author/Programmer
*1.0        09.05.2017   Orginal design                     Lukasz Janus

*****
*****/

#ifndef DEFINITON_H
#define DEFINITON_H

//definition from type - dane
#define n_first_name "$0"
#define n_last_name "&0"
#define n_gender      "!0"
#define n_parent      "r0"
#define n_day         "D0"
#define n_month       "D1"
#define n_year        "D2"
#define n_id_personaly "d0"
#define n_id_data     "d0"
#define n_id_relation  "d0"
#define n_children    "r1"
#define n_sibling     "r2"
#define n_grandparent "r3"
#define n_grandchildren "r4"
#define n_partner     "r5"
#define n_order       "r6"
#define n_gender_personaly "~0"
#define n_gender_date  "~1"
#define n_gender_relation "~2"
//definition from type - typ
#define t_first_name  36
#define t_last_name   38 //
#define t_gender      33
#define t_parent      114
#define t_day         68
#define t_month       69
#define t_year        70
#define t_id_personaly 148
#define t_id_data     149
#define t_id_relation  157//
#define t_children    115
#define t_sibling     116
#define t_grandparent 117
#define t_grandchildren 118

```



## *aplication.h*

```

/*****
*****
*"aplication.h"
*
*
*
*
*
*CONTENTS:
* "Klasa czysto abstrakcyjna dla klas poswieconych interfejsowi, klasa dziecko po
klasie enginer"
*HISTORY:
*version    Date Changes
*1.0        06.05.2017  Original design
                                           Author/Programmer
                                           Lukasz Witek vel Witkowski

*****/
#ifndef APLICATION_H
#define APLICATION_H
#include "../Enginer/enginer.h"
class C_aplication :
    public C_enginer
{
public:
    C_aplication();
    C_aplication(const C_aplication & aplication);
    C_aplication& operator=(const C_aplication& aplication);
    bool operator==(const C_aplication& aplication);
    bool operator!=(const C_aplication& aplication);
    //void m_add_human(C_human human);
    virtual ~C_aplication();
};
#endif // !APPLICATION_H

```

## *aplication\_txt.h*

```

/*****
*****
*"aplication_txt.h"
*
*
*
*
*
*CONTENTS:
* "Klasa dziecko po klasie C_aplication"
*HISTORY:
*version    Date Changes
*1.0        06.05.2017  Original design
                                           Author/Programmer
                                           Lukasz Witek vel Witkowski

```

*1.01	09.05.2017	Adding methods for menu	Mateusz Marchelewicz
*1.1	15.05.2017	Windows size changed	Mateusz Marchelewicz
*1.1	17.05.2017	Font size changed	Mateusz Marchelewicz
*1.2	21.05.2017	Menu modified, new methods added	Mateusz Marchelewicz
*1.2	22.05.2017	Menu modified	Mateusz Marchelewicz

```

*****
*****/
#ifndef APLICATION_TXT_H
#define APLICATION_TXT_H
#include "aplication.h"
#include <Windows.h>

using std::cout;
using std::endl;

class C_aplication_txt :
    public C_aplication
{
public:
    C_aplication_txt();
    C_aplication_txt(const C_aplication_txt & aplication_txt);
    C_aplication_txt& operator=(const C_aplication_txt& aplication_txt);
    bool operator==(const C_aplication_txt& aplication_txt);
    bool operator!=(const C_aplication_txt& aplication_txt);
    virtual ~C_aplication_txt();
    void SetWindow(int Width, int Height);
    void MainMenu();
    void Sub1();
    void SubMenu2();
    void ImportTree();
    void EditTree();
    void DisplayTree();
    void SearchTree();
    void CreateLogo();
    void CreateHuman();
};
#endif // !APPLICATION_TXT_H

```

## *string.h*

```
#ifndef N_STRIIING_H
#define N_STRIIING_H
#include <iostream>
#include <fstream>
class N_striing
{
private:
    char *Table = NULL;
    int size = NULL;
    int m_how_long(const char variable[]);
public:
    N_striing();
    N_striing(const char T[]);
    N_striing(const char &Gover);
    N_striing(const N_striing &C);
    N_striing& operator=(const N_striing &C);
    bool operator==(const N_striing &C);
    bool operator!=(const N_striing &C);
    bool operator>(N_striing &C);
    bool operator<(N_striing &C);
    N_striing operator+(const N_striing &c);
    N_striing operator+(const char &c);
    N_striing operator+(const char c[]);
    N_striing operator+(const int &i);
    N_striing& operator+=(const N_striing &c);
    N_striing& operator+=(const char &c);
    N_striing& operator+=(const char c[]);
    N_striing& operator+=(const int &i);
    char operator[](int values);
    const char* m_c_str();
    N_striing& m_itoa(long long i);
    long long m_atoi(int variable_start, int variable_stop);
    N_striing& m_push_back(const char &Gover); //dodaje na koniec znak*
    N_striing& m_push_back(const char Gover[]); //dodaje na koniec ciag znakow
    N_striing& m_push_front(const char &Gover); //dodaje na poczatek znak*
    N_striing& m_push_front(const char Gover[]); //dodaje na poczatek ciag
znakow
    N_striing& m_insert(int value, const char Gover); //wstawia w wybranym
miejsku znak na inny*
    N_striing& m_insert(int value, const char Gover[]); //wstawia w wybranym
miejsku ciag znakow
    N_striing& m_swap(const char &Gover_old, const char &Gover_new);
//podmienia na znak jesli znajdzie szukany (szukany, do wstawienia)*
    N_striing& m_swap(const char Gover_old[], const char Gover_new[]); //
podmieni na ciag znakow jezeli znajdzie szukany ciag znakow (szukany, do
wstawienia)
    N_striing& m_pop_back(); //usuwa ostatni znak
    N_striing& m_pop_front(); //usuwa pierwszy znak
    N_striing m_clear(); //czysci calosc
    N_striing& m_erase(int i); //usuwa krotke w "tablicy"
    N_striing& m_erase_ray(int value_front, int value_back); //usuwa wybrane
krotki w "tablicy"
```



```

        N_string& m_erase_ray(int value_front); //usuwa wszystkie krotki poczawszy
od podanej w "tablicy"
        N_string m_cut(int value_front, int value_back); //wycina wybrane krotki
(poczatek, koniec)
        bool m_wchat_char(const char &variable);
        bool m_wchat_char(const char variable[]);
        N_string m_cut(int value_front); //wycina od wybranej krotki do konca
        N_string& m_shift(int i, const char &value); //podmienia wybrana krotke na
podany znak
        int m_size(); //zwraca rozmiar
        friend std::ostream& operator<<(std::ostream &is, const N_string &C);
        friend N_string& operator >> (std::fstream &is, N_string &C);
        N_string& m_getline(std::ifstream &is);
        virtual ~N_string();
};
#endif // !N_string_H

```

## ***Vektor.h***

```

/*N_vektor();
N_vektor(int i);
N_vektor(const N_vektor &V);
N_vektor& operator=(const N_vektor &V);
bool operator==(const N_vektor &V);
T operator[](int values);
N_vektor& m_push_back(T inside);
N_vektor& m_push_front(T inside);
N_vektor& m_pop_back();
N_vektor& m_pop_front();
N_vektor& m_insert(int value, T inside);
N_vektor& m_erase(int value);
N_vektor& m_close();
int m_size();
~N_vektor(); */

#ifndef N_VEKTOR_H
#define N_VEKTOR_H
#include <iostream>
#include <cstdlib> //mozliwe ze nie potrzebne:p
template <typename T> class N_vektor
{
public:
    N_vektor() {
        Size = 0;
        Tab = new T[1];
    }
    N_vektor(int i) {
        if (i >= 1)
        {
            Size = i;
            Tab = new T[Size];
        }
        else
        {
            Size = 0; //mozna dodac wyjatek!
            Tab = new T[1];
        }
    }
};

```

```

    }
}
N_vektor(const N_vektor &V) {
    if (this != &V) *this = V;
}
N_vektor& operator=(const N_vektor &V) {
    if (this == &V) return *this;
    if (*this == V) return *this;
    if (Size == 0 || Tab == NULL)
    {
        this->~N_vektor();
        int i;
        Size = V.Size;
        this->Tab = new T[Size];
        for (i = 0; i < Size; i++)
        {
            this->Tab[i] = V.Tab[i];
        }
        return *this;
    }
    else
    {
        int i;
        this->m_close();
        Size = V.Size;
        this->Tab = new T[Size];
        for (i = 0; i < Size; i++)
        {
            this->Tab[i] = V.Tab[i];
        }
        return *this;
    }
}
bool operator==(const N_vektor &V) {
    if (this->Size == V.Size)
    {
        int i;
        for (i = 0; i < Size; i++)
        {
            if (Tab[i] != V.Tab[i]) return false;
        }
        return true;
    }
    return false;
}
bool operator!=(const N_vektor &V) {
    if (this->Size != V.Size)
    {
        int i;
        for (i = 0; i < Size; i++)
        {
            if (Tab[i] == V.Tab[i]) return true;
        }
        return false;
    }
    return true;
}
T operator[](int values)

```

```

        {
            return Tab[values];
        }
N_vektor& m_push_back(T inside) {
    int i;
    if (this->Size == 0 || this->Tab == NULL)
    {
        this->m_close();
        this->Size = 1;
        this->Tab = new T[Size];
        for (i = 0; i < Size; i++)
        {
            Tab[i] = inside;
        }
        return *this;
    }
    N_vektor C(*this);
    this->m_close();
    this->Size = 1 + C.Size;
    this->Tab = new T[Size];
    for (i = 0; i < Size - 1; i++)
    {
        Tab[i] = C.Tab[i];
    }
    Tab[Size - 1] = inside;
    return *this;
}
N_vektor& m_push_front(T inside) {
    int i;
    if (this->Size == 0 || this->Tab == NULL)
    {
        this->m_close();
        this->Size = 1;
        this->Tab = new T[Size];
        for (i = 0; i < Size; i++)
        {
            Tab[i] = inside;
        }
        return *this;
    }
    N_vektor C(*this);
    this->m_close();
    this->Size = 1 + C.Size;
    this->Tab = new T[Size];
    this->Tab[0] = inside;
    for (i = 0; i < C.Size; ++i)
    {
        this->Tab[(i + 1)] = C.Tab[i];
    }
    return *this;
}
N_vektor& m_pop_back() {
    m_erase(m_size() - 1);
    return *this;
}
N_vektor& m_pop_front() {
    m_erase(0);
    return *this;
}

```

```

    }
    N_vektor& m_insert(int value, T inside) {
        if (value >= 0 && value < Size)
        {
            int i, j = 0;;
            N_vektor STR(*this);
            this->m_close();
            Size = STR.Size;
            Tab = new T[(++Size)];
            for (i = 0; i < Size; i++)
            {
                if (i == value)
                {
                    Tab[i] = inside;
                    continue;
                }
                Tab[i] = STR.Tab[j];
                j++;
            }
        }
        return *this;
    }
    N_vektor& m_erase(int value) {
        if (value >= 0 && value < Size)
        {
            int j, i = 0;
            N_vektor C(*this);
            this->m_close();
            Size = C.Size;
            Tab = new T[(--Size)];
            for (j = 0; j < (Size + 1); j++)
            {
                if (j == value) continue;
                Tab[i] = C.Tab[j];
                i++;
            }
            return *this;
        }
        return *this;
    }
    N_vektor m_close() {
        N_vektor<T> V;
        this->~N_vektor();
        return V;
    }
    int m_size() { return Size;}
    virtual ~N_vektor() {
        if (Tab) {
            delete[] Tab;
            Tab = NULL;
            Size = NULL;
        }
    }
private:
    int Size = NULL;
    T* Tab = NULL;
};

```

```
#endif // !N_VEKTOR_H
```

**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.....	105
C_data .....	108
C_day .....	110
C_date .....	109
C_first_name .....	112
C_first_name .....	112
C_gender .....	113
C_id .....	118
C_id .....	118
C_last_name .....	119
C_month .....	119
C_date .....	109
C_year .....	125
C_date .....	109
C_element .....	111
C_tree .....	124
C_goverment .....	114
C_goverment_date .....	114
C_goverment_personaly.....	115
C_goverment_relation.....	115
C_human .....	117
C_relation .....	121
C_children .....	107
C_grandchildren .....	116
C_grandparents .....	116
C_parent .....	120
C_partner.....	120
C_sibling .....	122
C_save_load .....	121
C_sl_date.....	122
C_enginer.....	112
C_aplication.....	105
C_aplication_txt.....	106

C_sl_personalys .....	123
C_enginer .....	112
C_sl_relations.....	124
C_enginer .....	112
N_striing.....	125
N_vektor< T >.....	127
N_vektor< C_children >.....	127
N_vektor< C_date >.....	127
N_vektor< C_goverment_date >.....	127
N_vektor< C_goverment_personaly > .....	127
N_vektor< C_goverment_relation >.....	127
N_vektor< C_grandchildren >.....	127
N_vektor< C_grandparents > .....	127
N_vektor< C_human >.....	127
N_vektor< C_last_name > .....	127
N_vektor< C_parent >.....	127
N_vektor< C_sibling >.....	127
N_vektor< C_tree >.....	127

## Indeks klas

### *Lista klas*

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

<b>alfabet</b> .....	105
<b>C_aplication</b> .....	105
<b>C_aplication_txt</b> .....	106
<b>C_children</b> .....	107
<b>C_data</b> .....	108
<b>C_date</b> .....	109
<b>C_day</b> .....	110
<b>C_element</b> .....	111
<b>C_enginer</b> .....	112
<b>C_first_name</b> .....	112
<b>C_gender</b> .....	113
<b>C_goverment</b> .....	114
<b>C_goverment_date</b> .....	114
<b>C_goverment_personaly</b> .....	115
<b>C_goverment_relation</b> .....	115
<b>C_grandchildren</b> .....	116
<b>C_grandparents</b> .....	116
<b>C_human</b> .....	117
<b>C_id</b> .....	118
<b>C_last_name</b> .....	119
<b>C_month</b> .....	119
<b>C_parent</b> .....	120



<b>C_partner</b> .....	120
<b>C_relation</b> .....	121
<b>C_save_load</b> .....	121
<b>C_sibling</b> .....	122
<b>C_sl_date</b> .....	122
<b>C_sl_personalys</b> .....	123
<b>C_sl_relations</b> .....	124
<b>C_tree</b> .....	124
<b>C_year</b> .....	125
<b>N_striing</b> .....	125
<b>N_vektor&lt; T &gt;</b> .....	127

## 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`

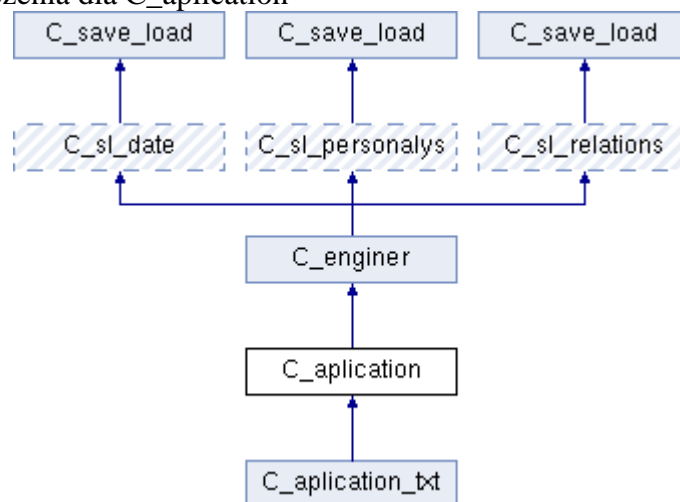
---

Dokumentacja dla tej struktury została wygenerowana z pliku:

7. `Data/Enginer/alphabet.h`

### *Dokumentacja klasy C\_aplication*

Diagram dziedziczenia dla C\_aplication



## 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)

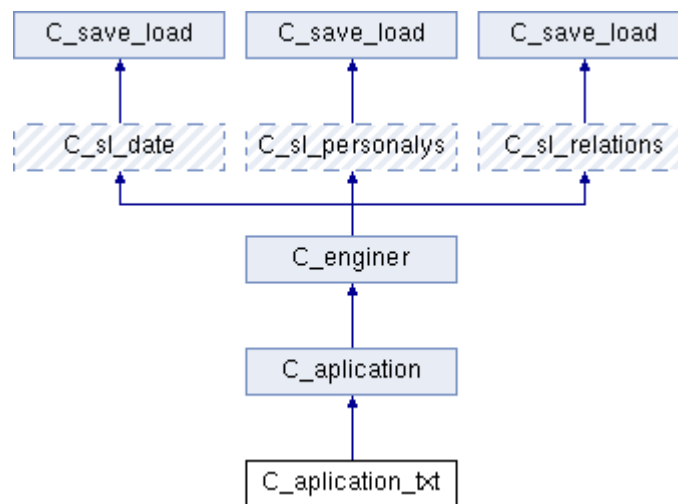
---

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** ()

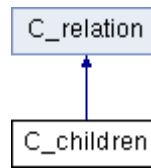
---

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)

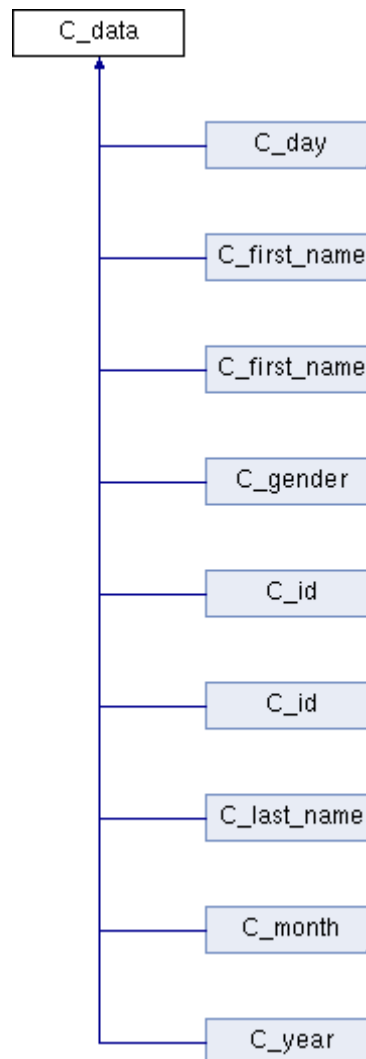
---

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)`

---

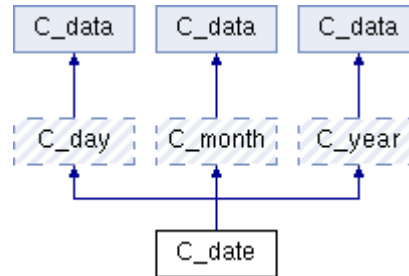
Dokumentacja dla tej klasy została wygenerowana z plików:

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

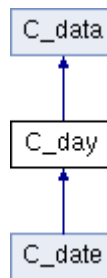
---

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

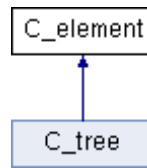
---

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)
```

---

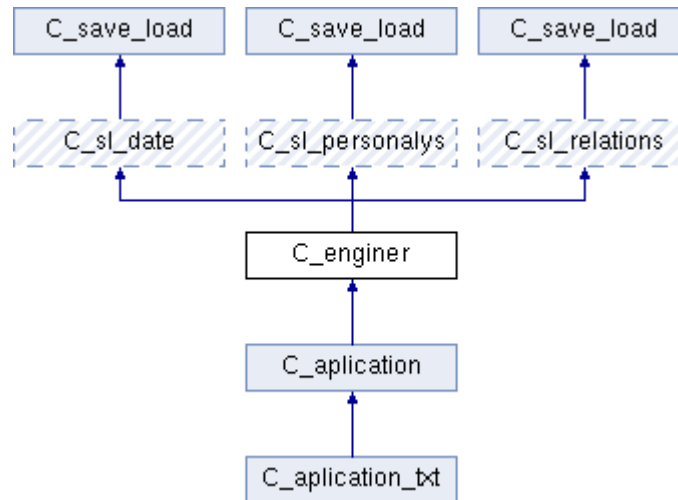
Dokumentacja dla tej klasy została wygenerowana z plików:

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)

---

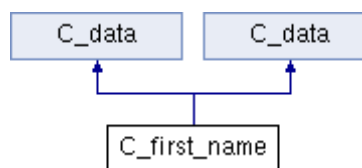
Dokumentacja dla tej klasy została wygenerowana z plików:

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)
```

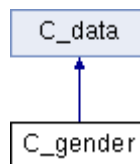
---

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)

---

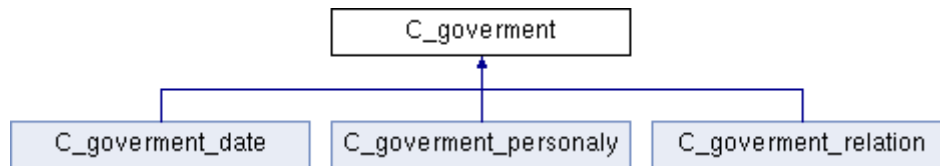
Dokumentacja dla tej klasy została wygenerowana z plików:

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

---

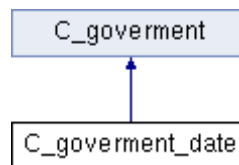
Dokumentacja dla tej klasy została wygenerowana z plików:

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 ()
```

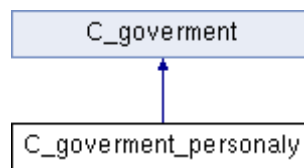
---

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 ()
```

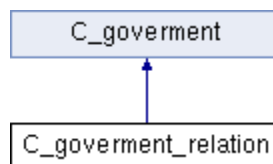
---

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:

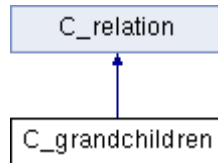
```

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:

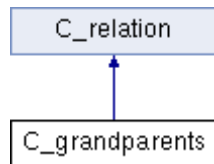
```

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:

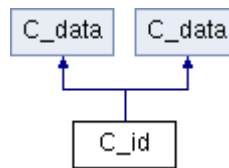
```

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:

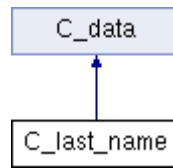
```

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)
```

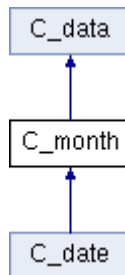
---

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
```

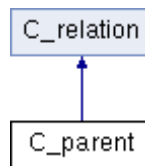
---

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)
```

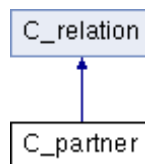
---

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)
```

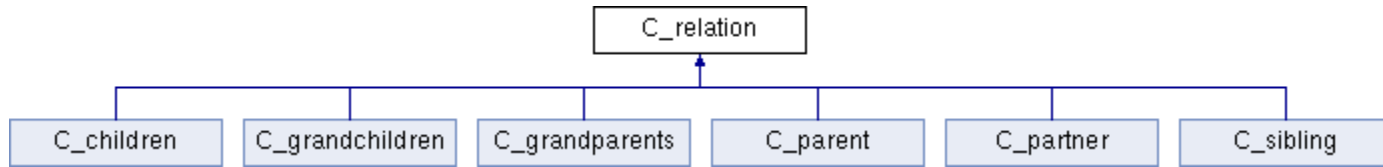
---

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)

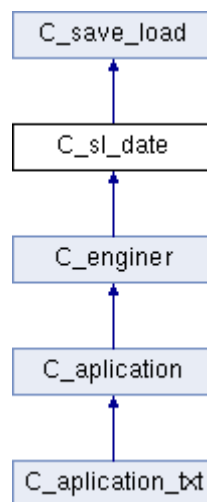
---

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:

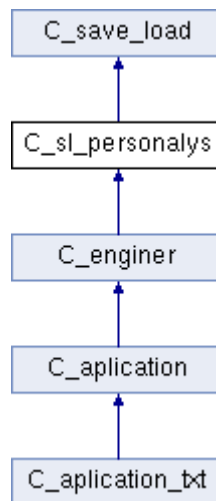
```

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)
```

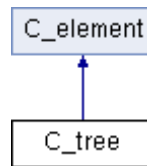
---

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 polskie znaki
101 C_first_name test101;
102 N_string fff = "Łukasz";
103 test101.m_get_contens(fff);
104 std::cout << "test 101:" << test101 << "\n";
105
106 C_last_name test102;
107 N_string fff2 = "Mikołaj";
108 test102.m_get_contens(fff2);
109 std::cout << "test 102:" << test102 << "\n";
110
111 C_gender test103;
112 //N_string fff3 = "Men";
113 //N_string fff3 = "men";
114 //N_string fff3 = "1";
115 //N_string fff3 = "true";
116 N_string fff3 = "Women";
117 //N_string fff3 = "women";
118 //N_string fff3 = "0";
119 //N_string 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**.

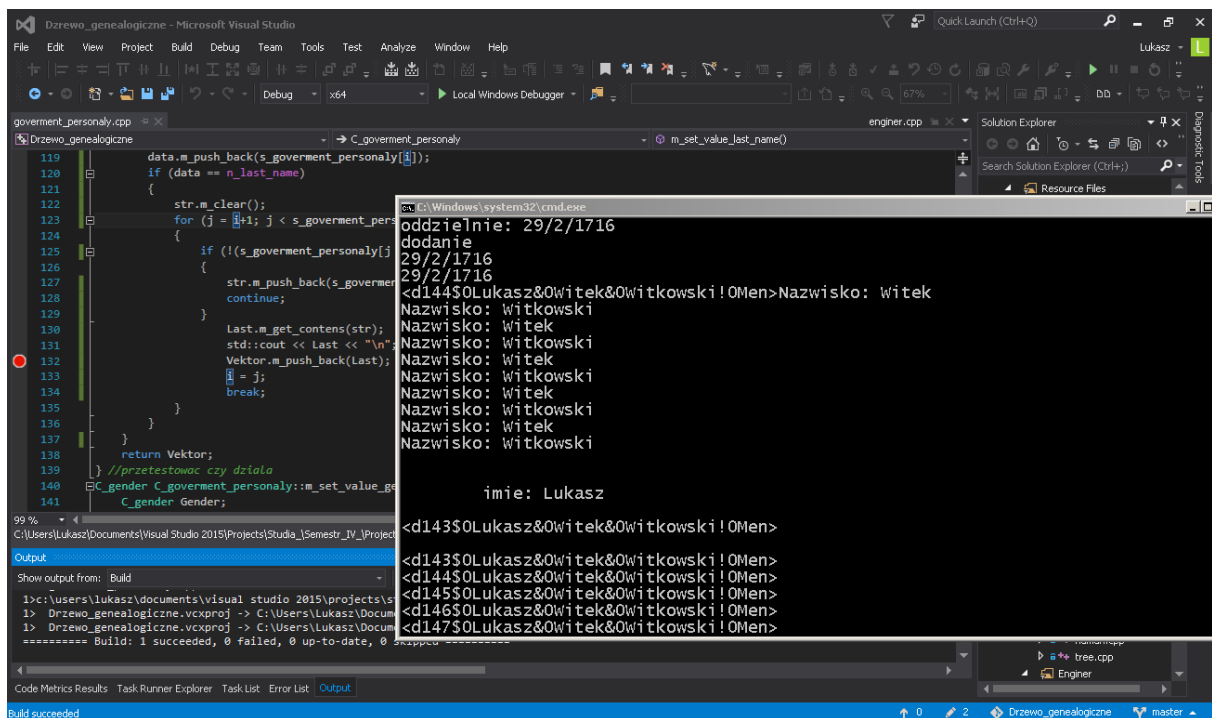
```
148
149
150 year2.m_get_year(yyy);
151 std::cout << yyy << "\n";
152 */
153
154 C_children chil;
155
156 chil.m_get_complete_content(34, 3);
157
```

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

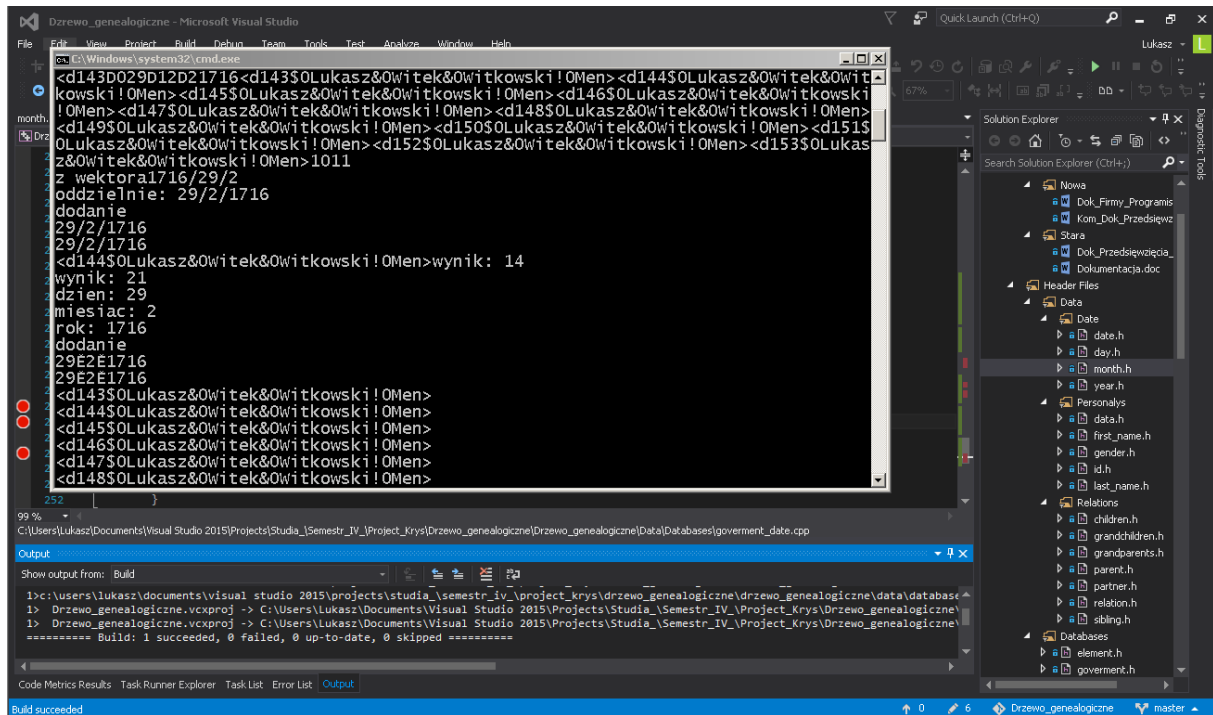
```
21 class C_relation
22 {
23 public:
24     C_relation();
25     C_relation(N_stringi reallt);
26     C_relation(const C_relation &relat);
27     C_relation& operator=(const C_relation &relat);
28     bool operator==(const C_relation &relat);
29     bool operator!=(const C_relation &relat);
30     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_stringi m_what_type();
34     virtual int m_set_variable() = 0;
35     virtual void m_get_complete_content(N_stringi 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.



Test na poprawne ładowanie danych z pliku będących interpretacją danych daty również zakończył się sukcesem.



```
<d143D029D12D21716<d143$OLukasz&OWitek&OWitkowski!OMen><d144$OLukasz&OWitek&OWitkowski!OMen><d145$OLukasz&OWitek&OWitkowski!OMen><d146$OLukasz&OWitek&OWitkowski!OMen><d147$OLukasz&OWitek&OWitkowski!OMen><d148$OLukasz&OWitek&OWitkowski!OMen><d149$OLukasz&OWitek&OWitkowski!OMen><d150$OLukasz&OWitek&OWitkowski!OMen><d151$OLukasz&OWitek&OWitkowski!OMen><d152$OLukasz&OWitek&OWitkowski!OMen><d153$OLukasz&OWitek&OWitkowski!OMen>1011
z wektora1716/29/2
oddzielnie: 29/2/1716
dodanie
29/2/1716
29/2/1716
<d144$OLukasz&OWitek&OWitkowski!OMen>wynik: 14
wynik: 21
dzień: 29
miesiąc: 2
rok: 1716
dodanie
29E2E1716
29E2E1716
<d143$OLukasz&OWitek&OWitkowski!OMen>
<d144$OLukasz&OWitek&OWitkowski!OMen>
<d145$OLukasz&OWitek&OWitkowski!OMen>
<d146$OLukasz&OWitek&OWitkowski!OMen>
<d147$OLukasz&OWitek&OWitkowski!OMen>
<d148$OLukasz&OWitek&OWitkowski!OMen>
}
99 %
C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_I_Semestr_IV_Project_Krys\Drzewo_genealogiczne\Data\Databases\government_date.cpp

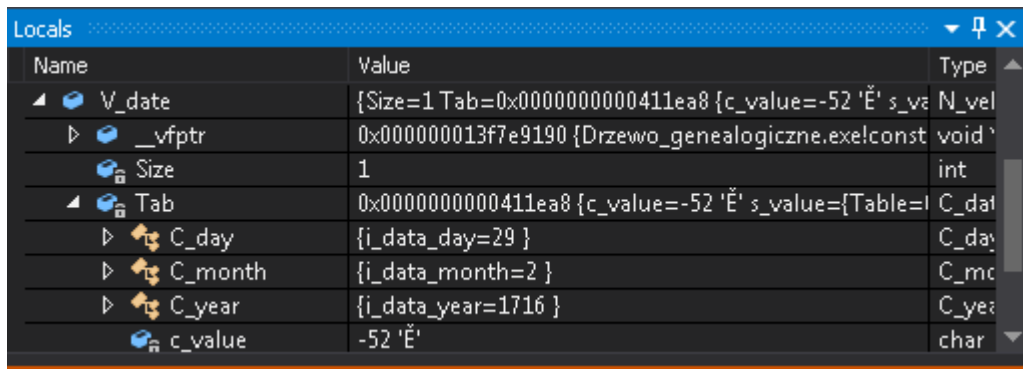
Output
Show output from: Build
1>c:\users\lukasz\documents\visual studio 2015\projects\studia_i_semestr_iv_project_krys\drzewo_genealogiczne\drzewo_genealogiczne\data\databases
1> Drzewo_genealogiczne.vcxproj -> C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_I_Semestr_IV_Project_Krys\Drzewo_genealogiczne
1> Drzewo_genealogiczne.vcxproj -> C:\Users\Lukasz\Documents\Visual Studio 2015\Projects\Studia_I_Semestr_IV_Project_Krys\Drzewo_genealogiczne
***** Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped *****

Code Metrics Results Task Runner Explorer Task List Error List Output
Build succeeded
```

Test na tworzenie humana i elementu. Wszystkie metody działają poprawnie i współpracują z klasami bazowymi.

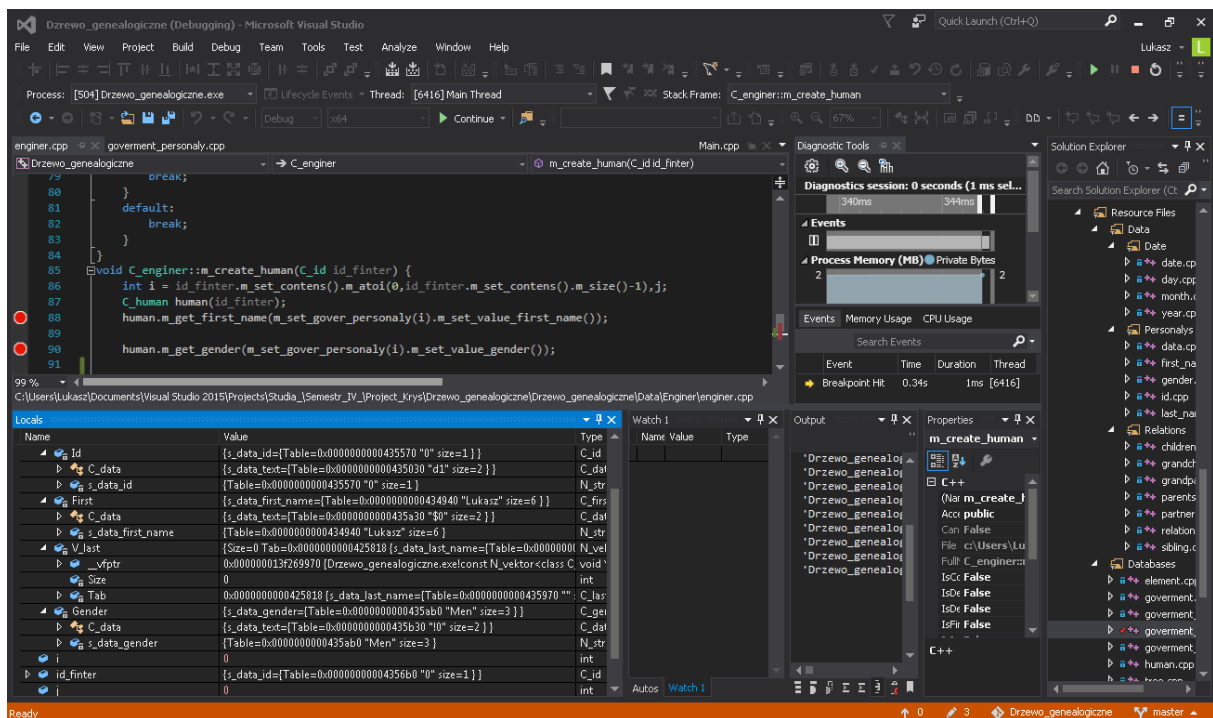
```
date13.m_get_type(data);
human.m_get_first_name(First);
human.m_get_last_name>Last);
human.m_get_last_name>Last1);
human.m_get_gender(gender);
human.m_get_date(date15[0]);
C_element element(human);
C_children children(human.m_set_id());
C_id id(10055);
C_grandchildren gchildren(human.m_set_id());
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 << "\nPrzedstawiony stworzony human:\n";
else std::cout << "\nblad z humanem:\n";
if (element == EEE) std::cout << "\nPrzedstawiony 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=0x0000000000411ea8 {c_value=-52 'Ě' s_value=N_vel	N_vel
__vfptr	0x000000013f7e9190 {Drzewo_genealogiczne.exe!const	void *
Size	1	int
Tab	0x0000000000411ea8 {c_value=-52 'Ě' s_value={Table=	C_da
C_day	{i_data_day=29 }	C_da
C_month	{i_data_month=2 }	C_mc
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. The function `m_create_human` is visible, which calls `human.m_get_first_name` and `human.m_get_gender`.
- Locals Window:** Shows the state of local variables. Key variables include:
  - `Id`: `{s_data_id={Table=0x0000000000435570 "0" size=1}}` (Type: `C_id`)
  - `C_data`: `{s_data_text={Table=0x0000000000435030 "d1" size=2}}` (Type: `C_da`)
  - `s_data_id`: `{Table=0x0000000000435570 "0" size=1}` (Type: `N_str`)
  - `First`: `{s_data_first_name={Table=0x0000000000434940 "Lukasz" size=6}}` (Type: `C_firs`)
  - `C_data`: `{s_data_text={Table=0x0000000000435a30 "0" size=2}}` (Type: `C_da`)
  - `s_data_first_name`: `{Table=0x0000000000434940 "Lukasz" size=6}` (Type: `N_str`)
  - `V_last`: `{Size=0 Tab=0x0000000000425818 {s_data_last_name={Table=0x00000000` (Type: `N_vel`)
  - `__vfptr`: `0x000000013f7e9970 {Drzewo_genealogiczne.exe!const N_vektor<class C` (Type: `void *`)
  - `Size`: `0` (Type: `int`)
  - `Tab`: `0x0000000000425818 {s_data_last_name={Table=0x0000000000435570 ""` (Type: `C_las`)
  - `Gender`: `{s_data_genders={Table=0x0000000000435ab0 "Men" size=3}}` (Type: `C_ge`)
  - `C_data`: `{s_data_text={Table=0x0000000000435b30 "0" size=2}}` (Type: `C_da`)
  - `s_data_gender`: `{Table=0x0000000000435ab0 "Men" size=3}` (Type: `N_str`)
  - `i`: `0` (Type: `int`)
  - `id_finter`: `{s_data_id={Table=0x00000000004356b0 "0" size=1}}` (Type: `C_id`)
  - `i`: `0` (Type: `int`)

- Solution Explorer:** Shows the project structure, including files like `data.cpp`, `day.cpp`, `month.cpp`, `year.cpp`, `personals`, `relations`, and `databases`.
- Diagnostic Tools:** Shows the `Events` window with a search bar and a table of events.
- Output Window:** Shows the output of the program, including the `m_create_human` function call.



## 7. HARMONOGRAM 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	wczytuje poprawnie
25.05.2017	ŁW	Test zapisu do pliku	zapisuje poprawnie
		Test wyświetlania drzewa	
		Test wyszukiwania osoby	
		Test działania relacji	
		Test menu + wszystkich podopcji	
		Testy końcowe	

## **8. DOKUMENTACJA ADMINISTRATORA**

### **Spis treści:**

1. Wstęp.
2. Ogólny opis programu.
3. Instalacja i usuwanie.
4. Konfiguracja programu.
5. Obsługa programu.
6. Zasady wypełniania pól i opis błędów.

### **1. Wstęp**

#### **1.1 Cel dokumentu**

Celem dokumentu jest przedstawienie funkcjonalności programu Genealogy Tree dla administratora systemu.

#### **1.2 Zakres dokumentu**

Dokument opisuje funkcjonalności administratora programu dostępne z poziomu aplikacji oraz pozostałe czynności administracyjne, wymagane do poprawnego działania programu.

## **2.    Ogólny opis programu**

Celem programu Genealogy Tree jest stworzenie podstawowej wersji drzewa genealogicznego. Program jest przeznaczony do użytkowania przez osoby indywidualne, rozpoczynające badania z zakresu genealogii.

Aktualna wersja programu pozwala m.in. na:

- stworzenie drzewa,
- nadanie mu nazwy (można stworzyć więcej, jak jedno drzewo),
- import/eksport pliku drzewa, stworzonego na innym komputerze,
- wyświetlenie drzewa w zakresie trzech pokoleń,
- manualne wprowadzanie i edytowanie informacji o osobach, wchodzącymi w skład drzewa,
- wyszukiwanie osób wchodzących w skład drzewa.

## **3.    Instalacja i usuwanie programu.**

Genealogy Tree jest programem jednostanowiskowym i nie przewiduje współdzielenia danych.

Program pracuje na komputerach klasy PC z zainstalowanym systemem Microsoft Windows.

### **3.1    Wymagania**

#### **Wymagania sprzętowe programu**

Konfiguracja minimalna:

- 

Konfiguracja zalecana:

-

Wymagania systemowe programu:

- Windows 2000 z Service Pack 4
- Windows XP z Service Pack 3
- Windows Vista z Service Pack 2
- Windows z Service Pack 1
- Windows 8/8.1
- Windows 10

z zainstalowanym aktualnym programem do obsługi archiwów \*.zip  
(zalecany program 7-Zip w wersji 16.04 lub wyższej).

### **3.2 Instalacja**

Instalacja Genealogy Tree przebiega identycznie niezależnie od wersji systemu Microsoft Windows. Przed instalacją należy sprawdzić, czy dysk **C:/** (lub inna lokalizacja, wskazana przez użytkownika) nie jest chroniony przed zapisem danych. W przypadku komunikatu o braku możliwości zapisu należy użyć uprawnień administratora lub wybrać inną lokalizację.

#### **3.2.1 Przebieg instalacji i uruchomienie programu:**

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 **C:/tree/genealogy\_inop.exe**.

#### **3.2.2. Usuwanie programu.**

Aby usunąć program:

1. Należy przejść do folderu, w którym jest zlokalizowany program.
2. Po zaznaczeniu folderu należy użyć kombinacji klawiszy Shift+Delete
3. Po pojawieniu się komunikatu „Czy na pewno chcesz usunąć ten folder?”, należy wybrać opcję „Tak”.

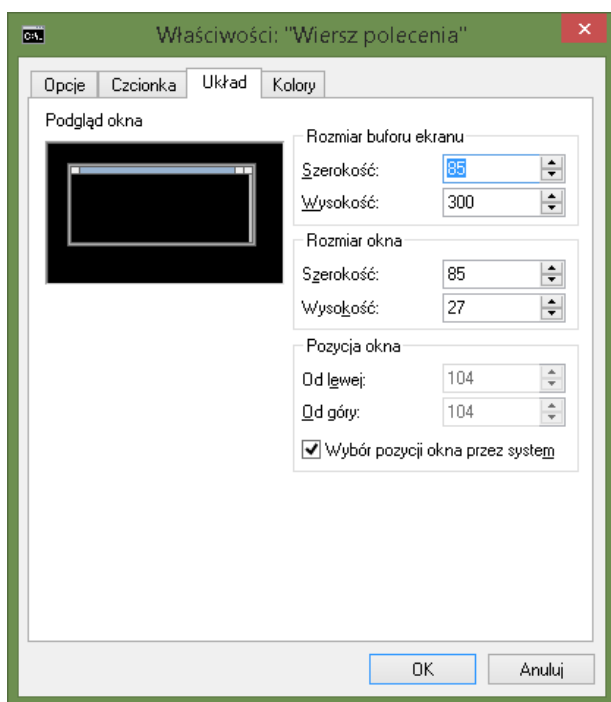
Operacja usunięcia programu jest nieodwracalna. Wraz z folderem głównym programu, usuwane są również pliki z zapisanymi drzewami.

## 4. Konfiguracja Programu.

### 4.1 Konfiguracja wyświetlania programu.

Jeśli po uruchomieniu programu Konsola System Windows nie dopasuje automatycznie rozmiaru okna, należy wykonać poniższe kroki:

1. W otwartym oknie konsoli należy kliknąć w lewy górny róg prawym przyciskiem myszy.
2. Po pojawieniu się menu kontekstowego należy wybrać opcję „Właściwości” i zakładkę „Układ”.
3. W zakładce „Układ” należy ręcznie zmienić ustawienia „Szerokość” zawartej w „Rozmiar Okna”, na rozmiar zapewniający komfort pracy użytkownika.
4. W niektórych przypadkach równolegle należy zmienić opcję „Szerokość” zawartą w „Rozmiar buforu ekranu” na wartość nie mniejszą, niż podaną w pkt. 3.



## 5. Obsługa programu

Program, po zainstalowaniu i ewentualnym skonfigurowaniu wyświetlacza, do obsługi wymaga jedynie klawiatury PC.

## 5.1 Obsługa programu – klawiatura PC.

Dostępne klawisze:

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

## 5.2 Kompletny schemat dostępnych funkcji programu:

### 5.2.1 Menu główne i podmenu: Create Tree, Load Tree

Jest to menu, z którego można wybrać tworzenie nowego drzewa (*Create Tree*) lub wyświetlenie (edycję) drzewa wcześniej stworzonego (*Load Tree*).

### 5.2.2 Tworzenie/wczytywanie nowego drzewa

Aby stworzyć nowe drzewo, należy wybrać opcję *Create Tree*, a następnie *Create New Tree*. Nazwa drzewa może składać się ze znaków anglojęzycznych i polskich znaków diakrytycznych.

W przypadku drzewa stworzonego na innym komputerze, należy użyć opcji *Import Tree*, następnie program poprosi o wpisanie nazwy stworzonego drzewa.

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

### 5.2.3 Menu Display Tree

Gdy drzewo zostało stworzone wcześniej, za pomocą opcji zawartych w *Display Tree* można wyświetlić jego zawartość:

- *Display from the oldest* – wyświetlenie pnia drzewa od najstarszego przedstawiciela rodziny
- *Search* – dwie opcje pozwalające wyszukiwać na podstawie danych takich, jak imię, nazwisko, data (dowolna, która występuje przy danej osobie)

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

### 5.2.4. Menu edycji drzewa

Menu *Edit Tree* pozwala otworzyć projekt drzewa, stworzony wcześniej w programie. Po wpisaniu nazwy drzewa, jest dostępny szereg opcji pozwalających na dodanie osoby (wraz z podstawowymi danymi), wyszukanie i edycja osoby, dodatnie relacji oraz zapisanie bieżącego drzewa.

<Nazwa drzewa>

1. Add a person
  - Add a first name, Add a surname, Add a gender, Add a date, Return
  - Add a marriage date (submenu → Add a person)
  - Add other relation: data (submenu → Add a person), → dodaj dzieci)
  - Add to (→ Find a person) as 'ascendent'/'descendent'
2. Find a person
  - Type a first name, Type a surname
3. Add a relation.
  - Find a person → Add a marriage date/Add other relation
4. Edit a relation.
  - Type a first name, Type a surname
5. Save a tree.
  - Save as
6. Exit.

### 5.3 Wyświetlanie pojedynczej osoby

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

- **narodziny** ( % )

- **ślub** ( \* )

- **zgon** ( + )

Przykładowy zapis:

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

## 6. Zasady wypełniania pól i opis błędów.

### 6.1 Format danych

Imiona i nazwiska można wprowadzać przy użyciu znaków anglojęzycznych i polskich znaków diakrytycznych. Daty należy podać w formacie ddmrrrrr.

### 6.2 Wprowadzanie danych i błędy

#### Wprowadzanie dat

**Komunikat: '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**

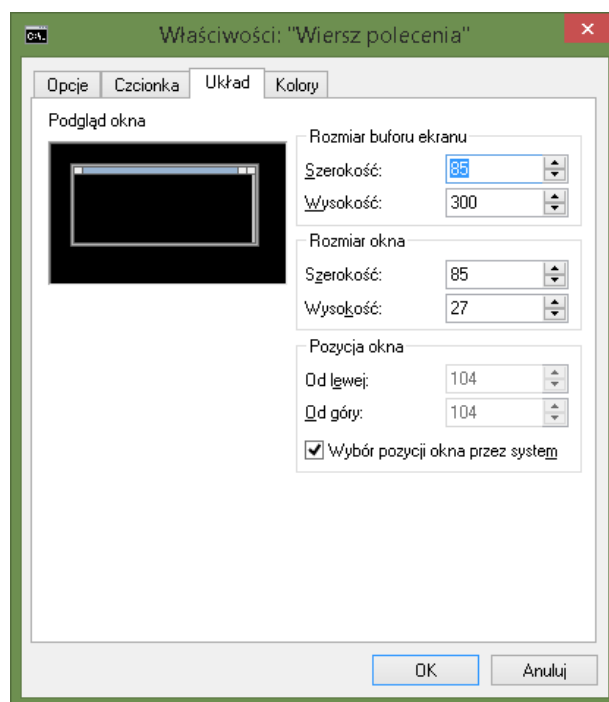
1. Display from the oldest – wyświetlanie drzewa od najstarszego przodka
2. Search – wyszukiwanie osób po danych personalnych lub na podstawie daty (np. data urodzenia, data ślubu)
  - a) Search by personal data
  - b) Search by date
  - c) Exit
3. Exit.

## 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:

<b>Jan Kowalski</b>  % 01.01.1900  + 02.02.1930	<b>Związki:</b> * 1919, Janina Nowak * 1921, Jadwiga Nowakowska
	<b>Relacje inne:</b> 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

→ Add a first name, Add a surname, Add a gender, Add a date,

Return

→ Add a marriage date (submenu → Add a person)

→ Add other relation: data (submenu → Add a person), → dodaj dzieci)

→ Add to (→ Find a person) as 'ascendent'/'descendent'

2. Find a person

→ Type a first name, Type a surname

3. Add a relation.

→ Find a person → Add a marriage date/Add other relation

4. Edit a relation.

→ Type a first name, Type a surname

5. Save a tree.

→ Save as

6. Exit.

## 10. RAPORT Z POSTĘPÓW PRODUKCJI OPROGRAMOWANIA (PROGRESS REPORT)

### 10.1. Tablica postępów

Document: **FamilyTree\_Documentation**

FFFFFF Program tworzący drzewo genealogiczne

Originator: z4 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)
<b>Platform</b>										
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		
<b>Displayed objects</b>										
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	Y	Y		
2.2.5.	[B4]	Saving data to files	Y	Y	N	N	Y	Y		
2.2.6.1	[B5]	Display a tree	Y	Y	Y	N	N	N		
2.2.6.2	[B5]	Search a person	Y	N	N	N	N	N		
2.2.6.3	[B5]	Relations working	N	N	N	N	N	N		
2.2.6.4	[B5]	Display a relations	N	N	N	N	N	N		
<b>Configura tion file</b>										
3.1.	[B1]	Using own string library	Y	Y	Y	Y	Y	Y		
3.2.	[B2]	Using own vector library	Y	Y	Y	Y	Y	Y		
3.3.1.	[B3]	Application name	Y	Y	Y	Y	Y	Y		
3.3.2.	[B3]	Application menu	Y	Y	N	Y	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		

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

## 11. DOKUMENTACJA QA (kontrola jakości)

### 11.1. Tabele QA

#### Lista testowanych klas:

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

**11.2. 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 wyrażeń 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 wyrażeń goto	Y	
Destruktor wirtualny	Y	
Użycie template'ów	N	nieobowiązkowe
Zaimplementowane metody	Y	
	podpis	
	Mateusz Marchelewicz	

## 12. DOKUMENTACJA TESTOWANIA

### 12.1. 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	

## 12.2. Raport błędów

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	
2.06	1.3	ŁW, MM	Błąd wyświetlania polskich znaków	2	ŁW	Źle zbudowane warunki w switchu
2.06	1.3	MM	Błąd przy wpisywaniu danych	1	ŁW	Nieprzeładowany operator >>



DATA	WERSJA KODU	TESTER	OPIS BŁĘDU	RODZA J BŁĘDU	OSOBA KORY- GUJĄCA	UWAGI

## 13. INNE