**ЗМІСТ**

Вступ ………………………………………………………………………….. 4

1 Аналіз технічного завдання ……………………………………………......5

2 Обгрунтування алгоритму й структури програми ………….………...…..6

3 Розробка програми…………………………………………………………. 8

3.1 Структура і опис класів …………………………………………..….9

3.2 Структура і опис методів …………………………………………...10

3.3 Створення об’єктів і розробка основної програми…. ……….…...11

4 Тестування програми і результати її виконання ……………………..…. 12

Висновки ………………………………………………………..………….... 14

Список використаних джерел ………………………………………….....…15

Додаток А . Текст програми............................................................................16

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

Реценз.

Н. Контр.

Затверд.

Зміст

Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

# ВСТУП

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

Реценз.

Н. Контр.

Затверд.

Вступ

Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

Актуальність програми для створення генеалогічного дерева на даний момент є незаперечною, тому що, сьогодні програмного забезпечення даного напрямку представлено досить мало. А тому і вибір у користувача невеликий. І якщо йому не подобається або інтерфейс, або функціональність конкретної програми то буде нелегко замінити її на якусь іншу, оскільки як вже зазначалося вище немає тієї широти вибору, яка б дозволяла кожному підібрати програмний продукт, який би повністю задовольняв його потреби. Також необхідність використання програм для створення і графічного відображення генеалогічних дерев полягає в тому, що вони дозволяють зокрема людині яка складає за їх допомогою дерево свого родоводу надовго зберігати дані про свій рід, графічно їх відображати і згодом через кілька років доповнювати та й зрештою передавати цю безперечно цінну інформацію майбутнім поколінням. Адже ми живемо в еру глобальної комп’ютеризації, в суспільстві де ПК стає невід’ємною часткою домашнього побуту. Тому і треба вчитися обробляти, зберігати всю потрібну в подальшому житті інформацію за допомогою комп’ютера. А програма для створення дерева родоводу безсумнівно набагато спрощує процес введення даних про членів роду, їх форматування, відображення, зберігання на довгі роки. В навчальному ж плані, створення такою програми необхідно для ґрунтовного вивчення технології об’єктно-орієнтованого програмування, освоєння роботи із динамічними структурами даних та засобів для роботи з файлами.

1. **АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ**

## В програмі передбачені такі можливості:

1. Поетапне створення нового генеалогічного дерева методом послідовного додавання кількох вузлів (даних про окремих людей);
2. Графічне відображення тільки створеного чи завантаженого із файлу дерева на екрані;
3. Зберігання новоствореного дерева у файл;
4. Завантаження раніше створеного дерева із файлу;
5. Редагування поточного дерева, шляхом зміни інформації конкретного вузла (особи);

## Специфікація програми:

## Введення даних в програму здійснюється через візуальні компоненти, такі як Edit, CheckBox, Button, ComboBox. Далі за допомогою відповідних властивостей цих компонентів дані зберігаються у потрібних полях даних класу з використанням конструктора.

## Вимоги до системи: IBM PC сумісний комп'ютер, 32-х або 64-х розрядна операційна система сімейства Microsoft Windows та середовище програмування Borland C++ Builder 6.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

5

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

.

Реценз.

Н. Контр.

Затверд.

Аналіз технічного завдання

Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

**2 ОБГРУНТУВАННЯ АЛГОРИТМУ І СТРУКТУРИ ПРОГРАМИ**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

6

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

.

Реценз.

Н. Контр.

Затверд.

Обгрунтування алгоритму і структури програми

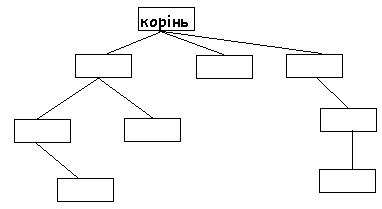
Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

Для реалізації завдання необхідно створити динамічну структуру типу «дерево», тобто спочатку потрібно створити корінь дерева (перший елемент) – вузол, який може містити вказівки на дочірні вузли. Причому обмежимося 5 дочірними елементами. В структурі типу «дерево» кожен його вузол крім першого має зв’язок із єдиним батьківським. Тобто в загальному «дерево» може мати наступний вигляд (рисунок 2.1):



**Рисунок 2.1 – структура «дерева»**

Пам’ять для вузлів «дерева» виділяється по мірі їх створення, динамічно, а при завершенні програми рекурсивно за допомогою деструктора звільняється. Рекурсивний обхід дерева можливий завдяки тому, що в даних кожного вузла зберігаються вказівники на дочірні вузли, за допомогою яких і здійснюється перехід з вищого рівня на нижчий і т.д. аж до кінцевого вузла, тобто такого який не містить жодного посилання на дочірні елементи.

Після створення «дерева» необхідно графічно відобразити його на екрані. Для цього скористаємося графічними можливостями Windows, які при роботі із графічними пристроями описуються за допомогою так званого дескриптора контексту пристрою (Device Context). Цей дескриптор дозволяє робити графічне

оформлення всіх видимих елементів у Windows – програмах за допомогою таких

інструментів як канва, пензлик, перо та шрифт. В C++ Builder ці інструменти реалізовані за допомогою наступних класів: TCanvas (для канви), **TBrush** (для пензля), **TPen** (для пера), **TFont** (для шрифту). Об’єкти цих класів створюються автоматично для кожної компоненти і стають доступними за допомогою відповідних властивостей: **Canvas**, **Brush**, **Pen**, **Font**, причому **Brush**, **Pen** і **Font** є підвластивостями властивості **Canvas**.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

7

ТНТУ КРКН 15.006.084.000 ПЗ

**Canvas** (канва, полотно) — це площина видимого елемента, на якій можна створювати інші графічні зображення.

**Pen** (перо) — це лінія, яка обрамляє зовнішній контур фігури (межа фігури).

**Brush** (пензель) — це внутрішня область фігури. За допомогою пензлика можна задавати колір зафарбування та стиль зафарбування, наприклад штриховими лініями.

**Font** — дозволяє вибрати шрифт, задати параметри шрифту. Виведення тексту здійснюється за допомогою методу **TextOut** властивості **Canvas**.

Зокрема в нашій програмі необхідними будуть такі функції для роботи з графікою:

Методи класу **TCanvas**:

**MoveTo(X,Y)** — переміщає перо в позицію (X,Y) не залишаючи слід;

**LineTo(X,Y)** — будує відрізок від поточної позиції до позиції (X,Y);

**TextOut(X, Y, Text)** — виводить рядок символів Text так, щоб верхній лівий кут прямокутника, що обрамляє текст мав координати (X,Y).

# РОЗРОБКА ПРОГРАМИ

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

8

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

Реценз.

Н. Контр.

Затверд.

Розробка програми

Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

* 1. **Структура і опис класів**

Для виконання поставленого завдання у програмі використовується клас **GenealTree** який містить в собі компоненти дані, та методи для їх опрацювання:

class GenealTree

{

public:

AnsiString Identificator;

AnsiString PIB;

bool is\_spouse;

AnsiString spouse\_PIB;

TDateTime birthday;

bool is\_dead;

TDateTime date\_of\_death;

unsigned int numb\_of\_children;

GenealTree \*child[5];

GenealTree(GenealTree\* rt);

~GenealTree();

void load\_idents\_to\_cb();

GenealTree\* find\_ident\_node(AnsiString f\_ident);

void load\_data\_to\_form();

};

Компоненти дані представляють зберігають інформацію по кожному вузлу «дерева», тобто такі дані про особу, як прізвище, ім’я, по-батькові, дата народження, дата смерті, для померлих людей, а також інформація про чоловіка

(жінку ) для того щоб можна було в одному вузлі зберігати як інформацію про

окрему людину так і про подружню пару. Також у класі описано поле даних

**Identificator** у якому зберігається специфічний ідентифікатор, що вказує на «рівень» (покоління) об’єкта даного класу, а також містить його порядковий номер на даному «рівні». Тобто для кореня (першого вузла) поле даних **Identificator** прийме значення 1\_1, а наприклад 3 його дочірні елементи – 2\_1, 2\_2, 2\_3 і т.д. Для того щоб достовірно знати чи у полі **date\_of\_death** міститься інформація про дату смерті у класі описане поле даних is\_dead типу **bool**, що може приймати значення true та false. Аналогічно для прийняття рішення, про відповідне опрацювання компоненту даних **spouse\_PIB** описане поле даних **is\_spouse**. І основним компонентом даних є масив із п’яти вказівників на описаний клас, описаний як:

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

9

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

Реценз.

Н. Контр.

Затверд.

Розробка програми

Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

**GenealTree \*child[5];**

Саме за допомогою цього масиву можливе рекурсивне проходження дерева, графічне виведення та збереження у файл.

Опис та реалізація методів класу винесені в окремий заголовочний файл Geneal\_tree\_class.h.

**3.2 Структура і опис методів**

Клас GenealTree має наступні методи:

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

10

ТНТУ КРКН 15.006.084.000 ПЗ

Конструктор **GenealTree(GenealTree \*rt=NULL)** за допомогою якого виконується ініціалізація об’єкта даного класу. Конструктор спрацьовує при успішному заповненні даними полів візуальних компонентів, розміщених на формі **frmProperities** та натисненні кнопки “OK”. Відбувається присвоєння компонентам даних класу відповідних значень введеним користувачем. Також конструктор має параметр, який якщо його не вказувати при виклику за замовчуванням приймає значення **NULL**. При створенні першого вузла «дерева» за допомогою, оператора умови if аналізується значення змінної **rt**, і якщо воно дорівнює **NULL** то це є ознакою того, що корінь «дерева» ще не створений, тобто не існує ще і самого «дерева», і конструктор був викликаний саме для створення першого, кореневого вузла. Тому полю **Identificator** зразу ж присвоюється значення 1\_1. В противному випадку у конструктор передається значення кореневого вузла, який вже безумовно не дорівнює нулю, а тому далі йде процес обрахування ідентифікатора для наступного вузла.

Деструктор **~GenealTree();** виконує звільнення зайнятої «деревом» динамічної пам’яті послідовно викликаючи деструктори для дочірних елементів, якщо вони існують.

**void load\_idents\_to\_cb();** - даний метод рекурсивно обходить всі вузли дерева і почергово додає значення їхнього поля даних **Identificator** до списку стрічок Items візуальної компоненти **ComboBox** яка відображає список наявних ідентифікаторів, при додаванні нового вузла в «дерево».

**GenealTree\* find\_ident\_node(AnsiString f\_ident);** - метод якому передається стрічка, що містить ідентифікатор, за значенням якого потрібно віднайти в дереві, вказівник на вузол, якому присвоєний даний ідентифікатор. Метод також рекурсивно обходить «дерево» і послідовно порівнює ідентифікатор кожного вузла із заданим у параметрі **f\_ident**.

**void load\_data\_to\_form(); -** метод виконує підготовку зовнішнього вигляду форми **frmProperities**, тобто завантажує у відповідні поля компонентів, що на ній розміщені дані із класу, для подальшого їх редагування. Метод викликається після натиснення на кнопку «Редагувати особу».

**3.3 Створення об’єктів і розробка основної програми**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

11

ТНТУ КРКН 15.006.084.000 ПЗ

У файлі MainFrm.cpp головної форми описано два об’єкти класу **GenealTree**: **root** і **current**. Перший з них служить для зберігання вказівника на корінь (перший елемент) «дерева». Оскільки майже рекурсивні методи відштовхуються від першого вузла, всі вони викликаються за допомогою даного вказівника. Вказівник **current** виконує роль, поточного вказівника, тобто вказує на один з вузлів «дерева», що розглядається.

У файлі головної форми описано такі функції, що не входять до методів класу:

v**oid draw\_tree() –** за допомогою якої здійснюється графічне відображення генеалогічного дерева з відповідним його форматуванням та встановленням візуальних зв’язків між вузлами «дерева»;

**void bt\_Add\_ElemClick(TObject \*Sender)** – викликається, коли користувач натискає кнопку «Додати особу». Виконує певні підготовчі дії над формою **frm\_properities** та викликає конструктор класу при натисканні користувачем кнопки «ОК» та коректного введення даних про особу;

**void bt\_Edit\_ElemClick(TObject \*Sender)** – викликається у відповідь на натискання користувачем кнопки «Редагувати особу»,

виконує деякі дії необхідні для подальшого виклику методів класу **load\_idents\_to\_cb()** та **load\_data\_to\_form()**;

**void bt\_Save\_TreeClick(TObject \*Sender)** – виконує збереження генеалогічного дерева у файл для зберігання і подальшого вікриття у програмі;

**void bt\_Load\_TreeClick(TObject \*Sender)** – забезпечує завантаження раніше створеного дерева родоводу у програму з подальшим його відображенням.

**4 ТЕСТУВАННЯ ПРОГРАМИ І РЕЗУЛБТАТИ ЇЇ ВИКОНАННЯ**

При запуску на виконання програми її вікно розгортається на весь екран. Основну його частину займає поле для графічного виведення генеалогічного дерева. Зверху на панелі розташовані кнопки «Додати особу» та «Редагувати особу», які відповідно виконують функції додавання нового вузла (особи чи подружньої пари) та редагування введених раніше даних.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

12

ТНТУ КРКН 15.006.084.000 ПЗ

Розроб.

Ферц О.С.

Перевір.

Бревус В.М.

Реценз.

Н. Контр.

Затверд.

**Тестування програми і результати її виконання**

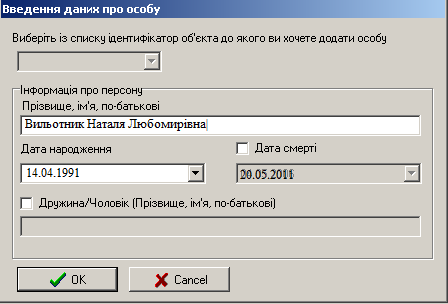
Літ.

Акрушів

29

**ТНТУ, ФІС, СН-21**

Для виведення на екрані зображення генеалогічного дерева, необхідно спочатку додати хоча б один вузол, тобто за допомогою кнопки «Додати особу», після натискання на яку з’явиться нове вікно (на рис. 4.1) у якому потрібно ввести необхідні дані про особу, такі як прізвище, ім’я, по-батькові, дату народження, для померлої людини дату смерті, а для одруженого чоловіка (заміжньої жінки) вказати прізвище, ім’я та по-батькові його жінки (чоловіка).



**Рисунок 4.1 – введення даних про особу**

Після коректного вводу даних про особу та натиснення кнопки «ОК», введені дані зразу ж будуть відображені на екрані. Після додавання першої особи

(предка), на формі, яка викликається кнопкою «Додати особу» стане доступним випадаючий, список, у якому можна буде вибрати ідентифікатор особи (пари), до якого слід додати нащадка (дитину).

Змн.

Арк.

№ докум.

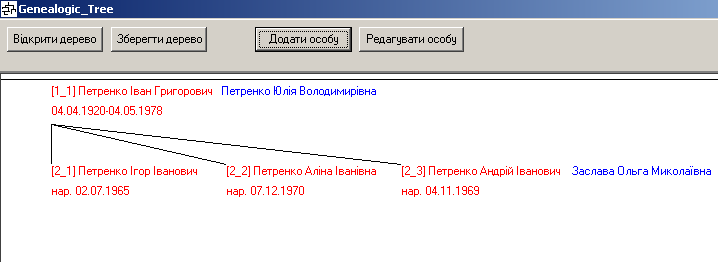
Підпис

Дата

Арк.

13

ТНТУ КРКН 15.006.084.000 ПЗ



**Рисунок 4.2 – дерево роду**

Оскільки програмою обмежена кількість дітей числом 5, то до одного і тої ж самої особи не вдасться додати більше 5-ох дітей. Нумерація ідентифікаторів є наступною: першому елементу дерева присвоєний ідентифікатор 1\_1, далі йтимуть 2\_1, 2\_2, .. , 2\_5. Нащадки ж елемента 2\_1 будуть 3\_1, .. ,3\_5, а для 2\_2 – 3\_6 .. 3\_10 і т.д. Наприклад створимо «дерево», що буде складатися з кореневого вузла і трьох дочірних. Графічне відображення генеалогічного дерева виглядатиме так (на рис. 4.2).

**ВИСНОВОК**

С++ - універсальна мова програмування. Основна її перевага над багатьма іншими в тому, що її можна використовувати для вирішення досить широкого спектру задач і не задумуватися над тим чи підтримує дана мова потрібну можливість чи ні. Хоча визнано, що С++ «важка» для вивчення мова, але той величезний потенціал який закладений в ній сповна виправдовує нелегкий курс засвоєння. А підтримка технології об’єктно-орієнтованого програмування дозволяє створювати компактні, легкі для розуміння, добре структуровані програми. Адже недарма майже всі сучасні мови програмування підтримують об’єктно-орієнтований підхід створення програм. Тепер можна легко змоделювати будь-який реальний об’єкт, виділивши найбільш істотні його ознаки та властивості, та написавши методи, що будуть проводити необхідні операції над ними.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

14

ТНТУ КРКН 15.006.084.000 ПЗ

На прикладі даної програми були використані переваги об’єктно-орієнтованого програмування а також вдалося ознайомитися із такими можливостями мови С++ та середовища програмування С++ Builder 6, як:

* + - використання графічних можливостей Windows;
    - використання класів для створення динамічних структур даних;
    - робота з бінарними файлами.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

## 1. Дж. Коплиен.  [Программирование на C++](http://progbook.net/ss/c-plus/3329-programmirovanie-na-c.html), 2005.

## 2. Либерти Д. [Освой самостоятельно C++ за 21 день](http://progbook.net/ss/c-plus/3552-osvoj-samostoyatelno-c-za-21-den.html),2006.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

15

ТНТУ КРКН 15.006.084.000 ПЗ

**КР КН 006.029.01.00 ПЗ**

## 3. В. Г. Давыдов. [Технологии программирования C++](http://progbook.net/ss/c-plus/3689-texnologii-programmirovaniya-c.html), 2005.