Gliwice, 01.02.2024

Programowanie I

**Projekt zaliczeniowy**

**"*Eksperyment z Win32*"**

**Kamil Bublij gr. 4/7**

1. **Opis projektu.**

Celem projektu było poznanie różnych technik programistycznych języka C++, w szczególności za pośrednictwem funkcjonalności dostępnych w środowisku Win32 i złożenie ich w spójną całość.

1. **Funkcjonalności.**

Zagadnienia poruszane w projekcie:

* elementy programowania obiektowego;
* biblioteka graficzna Direct2D, dokładniej DirectWrite;
* biblioteka graficzna GDI;
* dynamiczne zarządzanie pamięcią;
* kontenery (*map*, *set*, *string*, *vector*);
* wielowątkowość;
* szeroko rozumiane dialogi i kontrolki Win32;
* treści multimedialne (dźwięk, bitmapy);
* tworzenie przejrzystego interfejsu (menu, wstążki, skróty klawiszowe);
* weryfikacja danych wprowadzonych przez użytkownika;
* komendy preprocesora i konsolidatora;
* algorytmika (kodowania Huffmana).

1. **Przebieg realizacji.**

Program jest podzielony na mniejsze części – każda znacząco różniąca się funkcjonalnością część kodu ma osobny plik źródłowy i własny nagłówek.

**Wykaz plików:**

* *main.cpp, main.h* – główne pliki, zawierają podstawowe okno, instrukcje uruchamiania modułów i dialogu *O programie*;
* *Trigonometry.cpp, Trigonometry.h –* moduł rysujący funkcje okresowe;
* *GraphicsDemo.cpp, GraphicsDemo.h* – pobiera informacje   
  o możliwościach graficznych systemu użytkownika, po czym je wyświetla;
* *Tree.cpp, Tree.h* – pobiera informacje o tekście (lub parametrach kodowania), po czym na ich podstawie rysuje drzewo kodowania;
* *DWrite.cpp, DWrite.h –* pliki pomocnicze do obsługi biblioteki Direct2D;
* *Resources.rc, Resources.h –* pliki zasobów. Nie zawierają kodu, a elementy statyczne potrzebne programowi do działania;
* *tada.wav, (pliki osadzone w Resources.rc)* – pliki multimedialne utworzone przeze mnie, użyte za zgodą autora lub będące częścią domeny publicznej.

**Zastosowane algorytmy:**

W opisanych algorytmach będzie pojawiała się struktura zdefiniowana w kodzie jako *LEAF*:

TYPEDEF STRUCT LEAF{

UINT ID = NULL;

std::wstring tSymbol = {};

std::wstring tFPValue = {};

DOUBLE FPValue = 0;

UINT FFValue = 0;

STRUCT LEAF\* LeftChild = nullptr;

STRUCT LEAF\* RightChild = nullptr;

} LEAF, \* LPLEAF;

W dalszej części dokumentacji będę nazywał ją „liściem”.

* **tworzenie optymalnego drzewa kodowania Huffmana z kontenera liści**

Wstępne założenia: algorytm otrzymuje niepusty kontener, który zawiera wyłącznie liście. *Zaleca się*, żeby prawdopodobieństwa były znormalizowane (tutaj: suma *FPValue* po liściach równa 1), ale algorytm tego ani nie sprawdza, ani nie wymaga – inna część programu informuje użytkownika o problemie.

* 1. Jeśli w kontenerze pozostał jeden element, zakończ.
  2. Znajdź i wybierz dwa liście o najmniejszym prawdopodobieństwie.
  3. Utwórz nowy liść, który jako dzieci (tutaj: *LeftChild, RightChild*) będzie miał liście wybrane w kroku 2). Jego prawdopodobieństwo jest równe sumie prawdopodobieństw jego dzieci.
  4. Usuń liście z kroku 2) z listy.
  5. Umieść liść z kroku 3) w kontenerze.
  6. Wróć do 1).

W praktyce program tworzy drugą tablicę, która przechowuje częściowo przetworzone liście, a punkt 2) realizuje za pomocą sortowania tablicy   
i wybrania dwóch pierwszych elementów.

* **rysowanie drzewa binarnego**

Założenia: algorytm otrzymuje niepuste, **regularne** (każdy liść ma albo dwoje dzieci, albo żadnego) drzewo binarne.

1. Utwórz pusty stos przeznaczony na liście.
2. Ustaw się na najstarszym liściu (*korzeniu*).
3. Jeśli istnieje, narysuj lewe dziecko. Liść, na którym jesteś, wrzuć na stos z kroku 1) i ustaw się na jego lewym dziecku. Powtarzaj ten krok.
4. Dopóki jesteś prawym dzieckiem swojego rodzica, cofnij się do swojego rodzica (pierwszy element stosu z kroku 1)), po czym go usuń. Jeśli stos jest pusty, zakończ. Powtarzaj ten krok.
5. Trafiłeś do swojego rodzica z lewej strony, więc narysuj prawe dziecko i wrzuć je na stos z kroku 1). Wróć do kroku 3).

**Ważniejsze użyte funkcje bibliotek zewnętrznych:**

1. GDI:

**Pojęcia**: *kontekst urządzenia* – struktura przyznawana przez system w celu utworzenia jakiejś grafiki. Urządzeniem nie musi być monitor. Strukturę „obsadza się” obiektem (to **nie** obiekt jak w programowaniu obiektowym);   
*pędzel* – obiekt, którego biblioteka używa do rysowania grafiki (zawiera kolory, desenie itp.). Pędzel zna swoje współrzędne.

Jeśli nie podano inaczej, funkcje zwracają kod informujący o powodzeniu.

* + *Ellipse(), Rectangle()* – funkcje rysujące podstawowe kształty geometryczne. Przyjmują kontekst urządzenia i swoje wymiary.
  + *PolylineTo()* – funkcja rysująca ciągłą linię łamaną na podstawie podanych współrzędnych. Przyjmuje kontekst urządzenia, listę punktów i jej długość. Zmienia koordynaty pędzla.
  + *MoveToEx()* – zmienia współrzędne pędzla, niczego nie rysuje. Przyjmuje kontekst urządzenia, nowe współrzędne i opcjonalnie wskaźnik, gdzie funkcja zapisze, skąd się przesuwa.
  + *DrawTextEx()* – rysuje tekst. Przyjmuje kontekst urządzenia, wskaźnik do tekstu, jego długość, prostokąt, w którym będzie tekst i dwa argumenty ze szczegółami formatowania.

1. DirectWrite

**Pojęcia**: *fabryka* – obiekt generujący inne obiekty.

Jeśli nie podano inaczej, funkcje zwracają kod informujący o powodzeniu.

* *CreateTextFormat()* – tworzy parametry formatowania tekstu, które będą potem stosowane. Przyjmuje nazwę czcionki, opcjonalnie dodatkową nazwę kroju, informacje o: grubości, stylu (kursywa itp.), rozciągnięciu, rozmiar, język i wskaźnik do struktury, w której to zapisze.
* *DrawText()* – rysuje tekst. Przyjmuje wskaźnik do tekstu, długość do napisania, parametry formatowania, prostokąt, w którym ma być narysowany i pędzel.

1. **Instrukcja użytkownika.**

Program powinien być opatrzony ikoną, która, naturalnie, przedstawia kubek herbaty.

Użytkownik jest witany ekranem powitalnym, który mówi mu, co ma dalej zrobić.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Ku górze ekranu znajduje się menu, z którego można wybrać kilka pozycji:

*Notatka*: Zgodnie z konwencją, wszystkie pozycje menu zakończone wielokropkiem (…) powodują otwarcie nowego okna.

* Plik:
  + Zakończ (*Alt + F4*) – kończy program.
* Opcje:
  + Tryb szybki (*F4*) – na żądanie włącza/wyłącza użycie biblioteki Direct2D.
  + **<tylko skrót klawiszowy>** (*Shift + F4*) – niszczy klasę Direct2D, zmniejszając zużycie pamięci około 10-krotnie.
* „Fajne rzeczy”:
  + Właściwości grafiki… - otwiera opisany wcześniej moduł *GraphicsDemo*.
  + Pokaz trygonometrii… (*Ctrl + T*) – otwiera moduł *Trigonometry*.
  + Drzewko… (*Ctrl + H*) – otwiera moduł *Tree*.
* Pomoc:
  + O programie… (*F1*) – wyświetla informacje o projekcie.

**Moduł *GraphicsDemo***

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

Moduł nie zawiera menu. Okno reaguje na przewijanie paskiem i kółkiem myszy, przy zmianie rozmiaru paski dostosowują się odpowiednio.

**Moduł *Trigonometry***

Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie

Moduł zawiera klasyczne menu oraz *wstążkę*, która zawiera suwaki, służące do zmiany odpowiednio prędkości i przyspieszenia przesuwania się funkcji.

Wzór funkcji *Pokazowa 1*:

Wzór funkcji *Pokazowa 2*:

Struktura menu:

* Funkcje:

(wszystkie opcje odpowiadają za przełączanie wyświetlania funkcji)

* + - Sinus
    - Cosinus -
    - ~~Cocosinus~~ - :)
    - Tangens
    - Cotangens
    - Pokazowa 1
    - Pokazowa 2

**Dialog *O programie***

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, Czcionka

Opis wygenerowany automatycznie

Przy uruchomieniu powinna odegrać się dołączona z programem ścieżka dźwiękowa. Jeżeli jej nie ma, uruchomi się dźwięk jak przy systemowym komunikacie *Informacja*.

**5. Podsumowanie i wnioski.**

*W miejscu tym piszemy co zrealizowaliśmy, z czym były problemy. Ewentualnie jakie są dalsze kierunki rozwoju programu, czego nie udało się zrealizować*

Uwaga - do dokumentacji proszę nie wklejać całego kodu aplikacji. W sekcji realizacja można zmieścić fragmenty kodu, jeśli chcecie zwrócić uwagę na coś co było bardzo wymagające i jest warte głębszego omówienia.

Poza tym proszę komentować kod programu - to jest istotna część dokumentacji projektu.