Wstęp do programowania w języku C

Grupa TDr

Lista 9.

1. *(15 punktów w dzień 15 grudnia, później 10 punktów)*

Dane są następujące definicje.

```
const char* const xml_part =
   "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>";
const char* const opening_landscape_a4_svg =
   "<svg width=\"297mm\" height=\"210mm\" viewBox=\"0 0 297 210\" "
   "version=\"1.1\" xmlns=\"http://www.w3.org/2000/svg\">";
const char* const closing_svg =
   "</svg>";

void example() {
   FILE* ofile = fopen("example.svg", "w");
   fprintf(ofile, "%s\n%s\n", xml_part, opening_landscape_a4_svg);
   fprintf(ofile, "<polyline stroke=\"blue\" fill=\"none\" points=\"\n");
   fprintf(ofile,
      "    200.00, 72.00    140.00, 72.00\n"
      "    140.00, 72.00    110.00, 20.04\n"
      "    110.00, 20.04    80.00, 72.00\n"
      "    80.00, 72.00    20.00, 72.00\n");
   fprintf(ofile, "\"/>");
   fprintf(ofile, "%s\n", closing_svg);
   fclose(ofile);
}
```

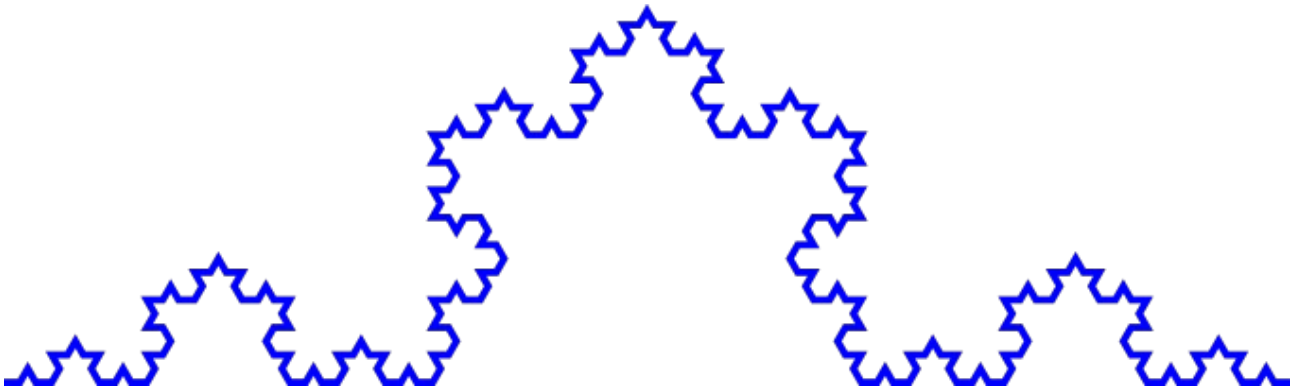
Wywołanie funkcji `example` tworzy plik o nazwie `example.svg` w formacie SVG o wymiarach poziomej kartki A4 i zapisuje na niej łamaną złożoną z czterech odcinków. Będzie go można obejrzeć w odpowiednim programie. Będzie to też plik tekstowy, z którym można poeksperymentować w notatniku.

Napisz 6 funkcji o podanych deklaracjach i wymaganiach.

1. `void Koch_points(FILE* ofile, int i, double x0, double y0, double x4, double y4);` ma do pliku `ofile` zapisać punkty `i`-tej iteracji krzywej Kocha o początku w punkcie (`x0`, `y0`) i końcu (`x4`, `y4`).

Jeśli przyjmiemy dokładność do dwóch miejsc po przecinku, to wywołanie `Koch_points(ofile, 1, 20, 72, 200, 72)` powinno wypisać punkty z przykładowej funkcji `example`. Zatem wygięcie krzywej powinno znajdować się po lewej jej stronie (uwaga: oś y układu współrzędnych SVG jest skierowana w dół, co zmienia jego orientację).

Więcej o krzywej Kocha można przeczytać np. na polskiej Wikipedii. Czwarta iteracja (otrzymana z wywołania `Koch_points(ofile, 4, 20, 72, 200, 72)`) tej krzywej wygląda zaś następująco (bez marginesów):



2. Funkcja `void example_content(FILE* ofile);` ma zapisać do danego jej pliku całą łamaną reprezentującą czwartą iterację krzywej Kocha (czyli od `"<polyline` do `\"/>"`).

3. Funkcja `void write_to_file(const char* filename, void script(FILE*));` ma otworzyć do zapisu plik o nazwie `filename`, zapisać do niego to, co zapisuje do swojego argumentu funkcja `script` i zamknąć otworzony plik.

Przetestuj ją zapisując do pliku o nazwie `test.txt` tekst `"Hello world!"`. Przetestują ją także funkcją `example_content`.

4. Funkcja `void write_a4_mm_svg(FILE* ofile, void write_content(FILE*));` ma zapisać do pliku deklarację formatu XML, otwierający znacznik `svg`, który zadeklaruje rozmiar poziomej strony A4 podany w milimetrach i ustanowi milimetry jednostką układu współrzędnych, oraz zamykający znacznik `svg` (czyli jak w przykładzie `example`), a między nimi umieści zawartość wypisywaną przez funkcję `write_content`.

5. Funkcja `void example_script(FILE* ofile);` ma za pomocą funkcji `write_a4_mm_svg` i `example_content` zapisać do danego pliku całą treść przykładowego pliku SVG przedstawiającego czwartą iterację krzywej Kocha.

6. Funkcja `void example1();` ma zapisać do pliku o nazwie `"example1.svg"` zawartość zdefiniowaną przez funkcję `example_script`.

2. *(15 punktów)*

w serwisie SKOS

Wstęp do programowania w języku C

Grupa TDr

Propozycje projektów programistycznych

Można zaproponować własny temat projektu, a do każdego projektu można zaproponować zmiany, własne rozszerzenia itp. Poniżej proponuję 10 przykładowych tematów projektów.

Informacje o tym, które tematy są zajęte, będę starał się umieszczać na mojej stronie.

1. Interpreter prostego języka programowania przez maszynę Créguta

Maszyna podana jest w tabeli 4. w tej pracy znajdującej się pod adresem https://link.springer.com/article/10.1007/s10990-007-9015-z (inna wersja podana jest w ogólnie dostępnym przeddruku: https://arxiv.org/pdf/2009.06984.pdf). Głównym problemem zdaje się być wydajna obsługa kopiowania struktur. Właściwie nie trzeba ich kopiować, bo wystarczy przekopiować wskaźnik, ale potrzeba liczyć, ile wskaźników jeszcze wskazuje na daną strukturę, żeby nie dochodziło do wycieków pamięci (jak w zadaniu 2. z listy 8.). Możliwych jest kilka rodzajów rozszerzeń.

2. Rozbudowany kalkulator wielomianów

Kalkulator wykonujący operacje na wielomianach jednej zmiennej. Możliwe rozszerzenia to obsługa wielu zmiennych, dzielenie wielomianów i branie reszty, a także obsługa unikodu itd.

3. Biblioteka do tworzenia rysunków w formacie SVG

Biblioteka powinna pozwalać na pisanie funkcji tworzących obrazy w formacie SVG. Oprócz obudowania rysowania np. prostokątów czy krzywych, należy stworzyć funkcje tworzące np. czytelne wykresy funkcji, diagramy itd.

angielska Wikipedia o SVG

4. Biblioteka do tworzenia dokumentów w formacie PDF

Zadanie podobne do SVG, ale chyba trochę trudniejsze ze względu na strukturę plików PDF i mniej dostępnych przykładów. Z tego względu wymaganych jest mniej funkcjonalności niż w przypadku SVG.

PDF Reference, version 1.7

5. Biblioteka do tworzenia i odczytywania zawężonego formatu ZIP

Biblioteka powinna pozwalać na tworzenie (niekoniecznie skompresowanych) folderów ZIP na podstawie zewnętrznych plików lub danych w programie i odczyt folderów takiego formatu. Najbardziej naturalnym rozszerzeniem jest oczywiście dodanie obsługi kompresji DEFLATE.

angielska Wikipedia o ZIPie

6. Biblioteka do tworzenia i odczytywania zawężonego formatu BMP

Biblioteka powinna pozwalać odczyt i modyfikację, a także tworzenie nowych plików BMP przy pewnych założeniach o ich formacie. Jako wybrany format najlepiej przyjąć ten, przy którym ma się największy dostęp do przykładowych plików BMP w tym formacie.

angielska Wikipedia o BMP

7. Narzędzie do wykorzystywania formularzy HTML jako projektów okien GTK

Narzędzie powinno umożliwiać wczytywanie formularzy w formacie HTML i tworzenie na ich podstawie okien GTK. Na podstawie atrybutu *onclick* i słownika w programie można podłączyć odpowiednie funkcje pod dane kontrolki.

8. Biblioteka do symulowania GTK w konsoli

Biblioteka powinna naśladować najczęściej używane funkcje wykorzystywane przez GTK tak, by mogła zastąpić odwołania prostych programów do GTK i wyświetlać znacznie uproszczony aktualny stan interfejsu w konsoli. Przy przyciskach wypisywać liczby, które użytkownik może podać na standardowe wejście, by symulować kliknięcie przycisku, a następnie wyświetlić kolejny stan itd.

9. Gry w GTK itp.

10. Narzędzie do gromadzenia informacji o drzewie filogenetycznym i ich wizualizacji np. przy użyciu programu GraphViz

Temat jest dosyć otwarty pod względem sposobu czerpania i przetwarzania informacji. Dane powinny zawierać informacje o szacowanych czasach rozgałęzień. Chodzi o to, żeby przygotować narzędzie do zaprezentowania czegoś ciekawego (dla laików) lub testowania hipotez.

Przykładowe dane: https://www.onezoom.org/life

Wstęp do programowania w języku C

Grupy TDr

Wyjściowy klucz oceniania projektu

Niezastępowalnymi częściami oceny projektu są:

- wybór tematu i propozycja klucza oceniania projektu do 5 I (można mejlowo, a najlepiej do 21 XII 2021 r.) *(10 punktów)*
- prezentacja postępów w projekcie do 12 I *(15 punktów)*
- prezentacja znacznych postępów do 26 I (można np. 12 I) *(15 punktów)*
- modularyzacja kodu za pomocą funkcji *(10 punktów)*
- ogólna zrozumiałość i poprawność kodu *(10 punktów)*

Zastępowalnymi częściami oceny projektu są:

- definicja i wykorzystanie odpowiednich typów danych *(10 punktów)*
- wykorzystywanie wprost dynamicznej alokacji pamięci *(10 punktów)*
- wykorzystanie kodowania UTF-8 *(10 punktów)*
- wykorzystanie operacji na plikach *(10 punktów)*
- wykorzystanie interfejsu okienkowego GTK+ *(10 punktów)*

Pozostałe 40 punktów ma odpowiadać aspektom specyficznym dla danego projektu.