Dokumentacja programu do generacji grafów

Autorzy: Adam Wieleba, Szymon Karbowiak

Środowisko: Program został napisany dla systemów z rodziny **Linux** oraz skompilowany za pomocą **GCC 7.5.0** (2017 Free Software Foundation, Inc.)

Ogólny opis działania:

Aplikacja służy do generacji grafów. Dostępne są trzy sposoby utworzenia grafu – dane mogą być:

1) podane przez użytkownika,

2) losowe,

3) podane przez czatbota.

Program ma opcję wczytywania poleceń i wypisania grafu do pliku tekstowego (domyślnie jest to wejście i wyjście standardowe).

Szczegółowy opis działania:

W celu uruchomienia programu należy wykonać plik „jimp_proj.exe”. Aplikacja powita użytkownika wiadomością:

Chcesz utworzyć graf samemu czy przy pomocy czatbota?
[s - samemu, c - czatbot]

Użytkownik musi wybrać opcję podając jedną z liter "s" lub "c"

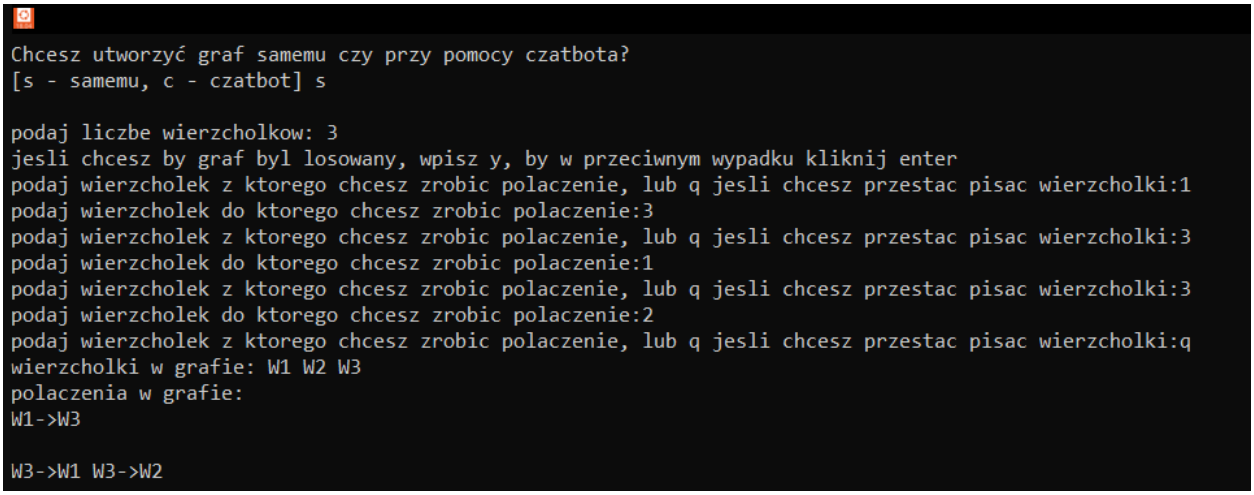
1) Generacja grafu bez pomocy czatbota

- Po wykonaniu poprzedniego kroku program odpowie zapytaniem "podaj liczbe wierzcholkow: ", na które trzeba odpowiedzieć podając liczbę całkowitą. Pojawi się wtedy kolejne zapytanie "jesli chcesz by graf byl losowany, wpisz y, by w przeciwnym wypadku kliknij enter ", na które ponownie należy odpowiedzieć zgodnie z instrukcją. Jeśli zostanie wybrana generacja losowa, połączenia między wierzchołkami utworzą się w sposób losowy, a program wypisze wynik i zakończy działanie. W przeciwnym wypadku, użytkownik będzie proszony o podanie połączeń występujących w grafie za pomocą naprzemiennych wiadomości "podaj wierzcholek z ktorego chcesz zrobic polaczenie, lub

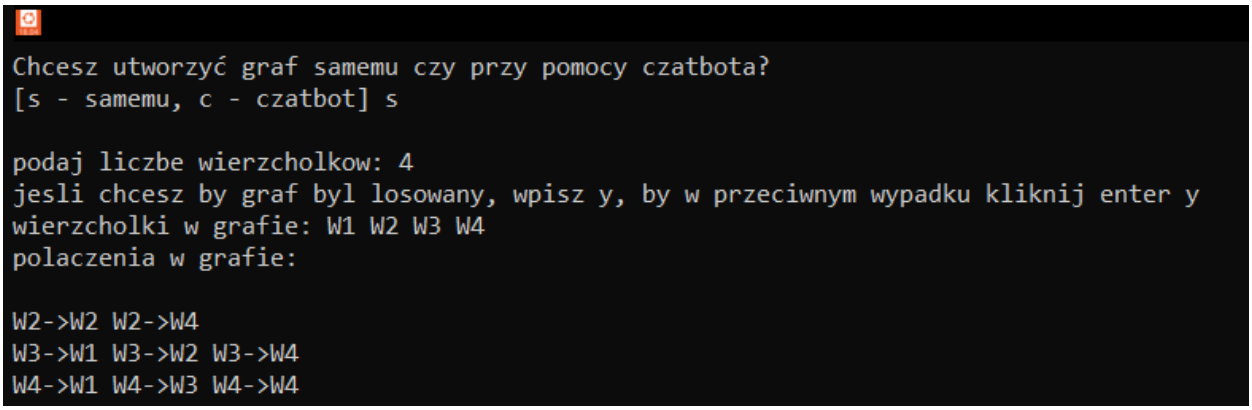
q jesli chcesz przestac pisac wierzcholki:" oraz "podaj wierzcholek do ktorego chcesz zrobic polaczenie:". Na przykład: odpowiedzenie na pierwszą wiadomość „1”, a na drugą „2”, a następnie w ten sam sposób „4”, „3” i „q” będzie skutkować utworzeniem połączeń W1 => W2 oraz W4 => W3. Po wydaniu polecenia o zaprzestaniu dalszego podawania wierzchołków, program wypisze wynik i zakończy działanie.

Przykłady:

(a) Generacja na podstawie danych od użytkownika



(b) Generacja losowa

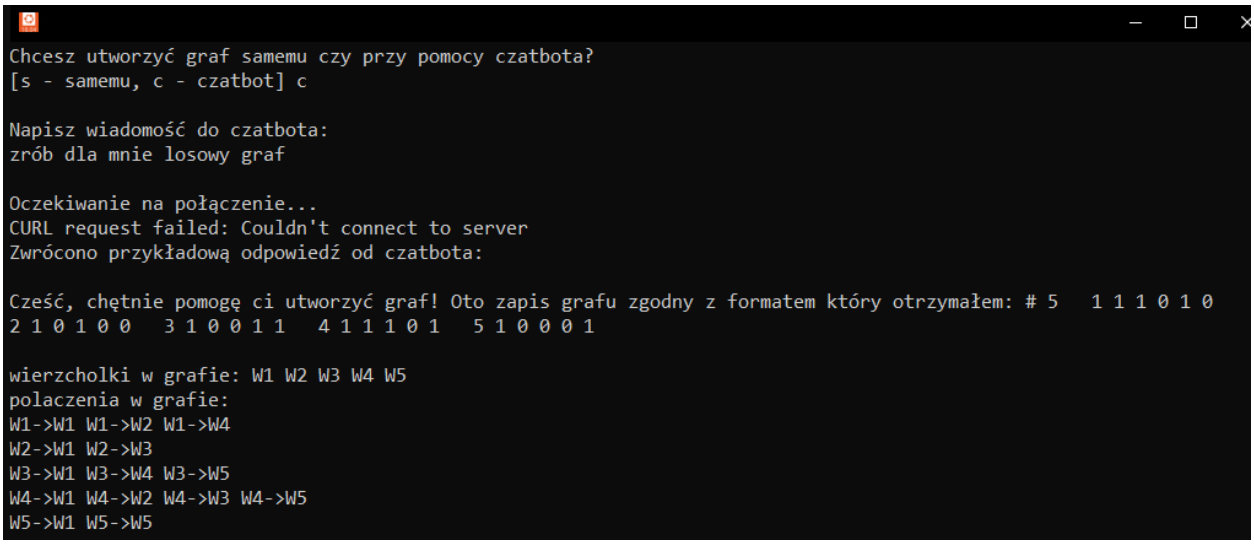


2) Generacja grafu przy użyciu czatbota

- Jeśli użytkownik zdecyduje się skorzystać z pomocy czatbota, zostanie poproszony o napisanie wiadomości ("Napisz wiadomość do czatbota: ”), w której może przekazać swoje preferencje np. „niech graf będzie miał 4 wierzchołki” albo „zrób graf w którym będzie 5 połączeń”. Ta wiadomość zostanie następnie wysłana na serwer, a program odbierze

odpowiedź i przeszuka ją pod względem zawartości w poszukiwaniu opisu grafu w formacie „# 2 1 0 0 2 1 1”, którego wyjaśnienie znajduje się w sekcji „Funkcje” w zakładce „ExtractData”. Jeśli nie uda się połączyć z serwerem, odpowiedź od czatbota zostanie ustawiona na przykładową wiadomość zdefiniowaną wewnętrznie. W obu przypadkach na podstawie otrzymanych danych program wypisze wynik i zakończy działanie.

Przykład:



Wynikiem działania programu jest opis tekstowy grafu w następującej postaci:

wierzcholki w grafie: W1 W2 W3

polaczenia w grafie:

W1->W1 W1->W2 W1->W3

W2->W3

W3->W1 W3->W2

gdzie po „wierzcholki w grafie: ” są wypisane wszystkie wierzchołki, a pod „polaczenia w grafie: ” wszystkie połączenia wychodzące z każdego wierzchołka po kolei (jeśli z danego wierzchołka nie wychodzi żadne połączenie, będzie w tym miejscu pusta linia tekstu).

Wynik jest domyślnie wypisywany na wyjście standardowe (stdout), jednak przy uruchamianiu można dodać **parametr wejściowy " -wf ", który spowoduje że wynik zostanie zapisany do pliku output.txt**, a na wyjście standardowe zostanie jedynie wypisana wiadomość „Graf został zapisany do pliku output.txt”. Analogicznie istnieje również opcja wywołania **" -rf ",** po użyciu której **polecenia będą odczytywane z pliku**

input.txt. Aby program właściwie odczytał instrukcje z pliku, należy w kolejnych liniach tekstu umieścić odpowiedź na każde poszczególne zapytanie programu (wyjątkiem są połączenia wierzchołków,). Przykład:

s
3

1 1 1 2 3 1
q

skutkuje
(1) przejściem do trybu samodzielnej pracy „ **s** ”
(2) ustawieniem liczby wierzchołków równej trzy „ **3** ”
(3) odrzuceniem propozycji utworzenia losowych połączeń „[enter ozn. jako **pusta linia**]”
(4) utworzeniem połączeń W1->W1, W1->W2, W3->W1 „ **1 1 1 2 3 1** ”
(5) zakończeniem podawania danych „ **q** ” (a następnie zwróceniem wyniku i zakończeniem programu)

Dodatkowe uwagi:
- instrukcja „enter” w pliku tekstowym powinna być oznaczona jako pusta linia, tak jak na powyższym przykładzie
- kolejność podania opcji nie ma znaczenia (tzn. może być zarówno -rf -wf oraz -wf -rf)

Funkcje

main:

- główne menu programu, wywoływanie reszty funkcji w programie, tworzenie grafu bez pomocy czata (według schematu w sekcji Szczegółowy opis działania)

WypiszGraf:

- wypisuje graf na podane wyjście

- aby wypisać graf, zwyczajne przechodzi przez tablicę reprezentującą go

PodajWynik:

- wywołuje funkcję WypiszGraf z odpowiednimi parametrami, m.in. strumieniem wyjściowym do wypisania grafu

- używa tylko podstawowych funkcji z języka C oraz funkcji WypiszGraf

AskChatbot:

- obsługuje połączenie z serwerem, na którym znajduje się czatbot

- postępowanie zgodne z czterema kolejnymi etapami:

1) nawiązuje połączenie,

2) wysyła podaną przez użytkownika wiadomość,

3) zwraca otrzymaną od czatbota odpowiedź,

4) zamyka połączenie.

Wszystkie te operacje są wykonywane za pomocą biblioteki curl (skrót od: Client URL), a wiadomości są przekazywane w plikach JSON

- funkcja zwraca dynamiczny ciąg znaków (tzn z dynamicznie przypisaną na niego pamięcią), zawierający odpowiedź otrzymaną od czatbota lub wiadomość przykładową jeśli nie uda się połączyć z serwerem z przyczyn zewnętrznych

ExtractData:

- przyjmuje ciąg znaków, w którym spodziewamy się opisu grafu w następującym formacie:

"# 3 1 0 0 1 2 1 1 0 3 1 0 1",

gdzie # oznacza początek danych dotyczących grafu, występująca po nim liczba to ilość wierzchołków, następnie dla każdego wierzchołka występuje ciąg liczbowy w postaci [nr wierzchołka] [0/1] [0/1] ... [0/1] (cyfr po numerze wierzchołka jest tyle ile wszystkich wierzchołków, "1" na n-tym miejscu oznacza że jest połączenie do wierzchołka Wn, "0" - nie ma połącznie).

Zatem dla podanego przykładu "# 3 **1** 0 0 1 **2** 1 1 0 **3** 1 0 1" interpretacja jest następująca:

1) "#" => początek informacji o grafie

2) "3" => ilość wierzchołków = 3

3) "**1** 0 0 1" => [**wierzchołek nr 1**] [brak połączenia do W1] [brak połączenia do W2] [ma połączenie do W3]

analogicznie:

4) "**2** 1 1 0" => [**W2**] [ma połączenie do W1] [ma połączenie do W2] [brak połączenia do W3]

5) "**3** 1 0 1" => [**W3**] [ma połączenie do W1] [brak połączenia do W2] [ma połączenie do W3]

- otrzymany ciąg znaków musi być dynamiczny, ponieważ do pozyskania danych z otrzymanego tekstu używana jest funkcja strtok (z biblioteki string.h). Dzięki niej możliwa jest tokenizacja danego ciągu znaków i wydobywanie tylko tych tokenów, które zawierają interesujące nas informacje.

Inne szczegóły techniczne:

- opis grafu jest przechowywany w dwuwymiarowej tablicy zaalokowanej dynamicznie

Ograniczenia:

- Aplikacja została napisana dla systemów z rodziny Linux, dlatego jej podstawowa wersja nie działa na popularnym systemie operacyjnym Windows.

- **Nie zastosowanie się w odpowiedni sposób do poleceń programu skutkuje wypisaniem odpowiedniego komunikatu oraz kontrolowanym zakończeniem jego działania.**