Politechnika Śląska

Wydział Informatyki, Elektroniki i Informatyki

**Podstawy Programowania Komputerów**

Temat projektu: TUC

|  |  |
| --- | --- |
| autor | Hubert Olszewski |
| prowadzący | Mgr inż. Dariusz Marek |
| rok akademicki | 2020/2021 |
| kierunek | informatyka |
| rodzaj studiów | SSI |
| semestr | 3 |

|  |  |
| --- | --- |
| **2** | **Hubert Olszewski** |

|  |  |
| --- | --- |
| **1 Treść zadania** | **3** |

**1 Treść zadania**

Napisać program symulujący działanie układu złożonego z bramek logicznych. Dostępne są następujące bramki: **and**, **nand**, **or**, **nor**, **xor**, **xnor**, **neg**. Każda bramka ma jedno wyjście i dwa wejścia. Jedynym wyjątkiem jest bramka **neg**, która ma jedno wejście i jedno wyjście. Połączenie wejść i wyjść bramek jest traktowane jako węzeł. Plik wejściowy przedstawiający układ ma następujący format: W pierwszej linii podane są numery węzłów będących wejściem układu. W drugiej linii numery węzłów będące wyjściem układu. Każda następna linia zawiera opis jednej bramki w postaci:

<węzeł wejściowy> <węzeł wejściowy> <węzeł wyjściowy>

Drugi plik wejściowy zawiera w każdej linii stany wejść, dla których należy znaleźć stan wyjść.

Plik wynikowy podaje wartości wyjść dla zadanych stanów wejść.

.**2 Analiza zadania**

Zagadnienie przedstawia problem dodawania odczytanych danych z plików tekstowych do odpowiadających im struktur i wygenerowanie na tej podstawie pliku z rozwiązaniem układu.

**2.1 Struktury danych**

W programie wykorzystano kontenery **STL**, wskaźniki inteligentne typu **shared\_ptr** oraz strukturę klas reprezentujące bramki logiczne. W programie pojawia się również strukturowa klasowa reprezentująca stany logiczne.

**2.2 Algorytmy**

Program wykorzystuje vectory przechowujące dwa typy danych. Pierwszy vector o typie wskaźnika na klasę bazową, przechowuje obiekty klas pochodnych reprezentujące bramki logiczne. Drugi vector o typie klasy reprezentującej stany logiczne, przechowuje odczytane stany wejściowe. Wypisanie elementów jest realizowane iteracyjnie poprzez przejście po obiektach vectora. Zwolnienie pamięci jest zrealizowane w sposób automatyczny, dzięki wykorzystaniu wskaźników inteligentnych, które realizują te zagadnienie we własnym zakresie.

**3 Specyfikacja zewnętrzna**

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników dowolna):

**-u** plik wejściowy z układem

**-i** plik wejściowy ze stanami wejść

**-o** plik wyjściowy ze stanami wyjść

**-h** wyświetlenie wszystkich możliwych przełączników oraz krótką instrukcje

Uruchomienie programu bez parametrów powoduje wypisanie tej samej instrukcji, co w przypadku uruchomienia programu z przełącznikiem **–h**.

**4 Specyfikacja wewnętrzna**

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs od logiki aplikacji.

**4.1 Ogólna struktura programu**

W funkcji głównej wywołana jest funkcja **readParams**. Funkcja ta sprawdza, czy program został wywołany w prawidłowy sposób. Gdy program nie został wywołany prawidłowo, zostaje zwrócona odpowiednia wartość, symbolizująca rodzaj błędu, która jest przekazywana do funkcji **printErrorGates**. Funkcja ta wypisuje stosowany komunikat i program się kończy. Następnie wywoływana jest metoda **loadGates**.Metoda ta otwiera plik wejściowy, sczytuje dane i umieszcza je do vectora. Po sczytaniu wszystkich danych tekstowych funkcja zamyka plik. W razie wystąpienia błędu metoda zwraca odpowiednią wartość, jak w przypadku omówionym wyżej, do funkcji **printErrorGates**, która to wypisuje stosowny komunikat i kończy działanie programu. Następnie wywoływana jest metoda **loadStates**. Metoda ta postępuje w bardzo podobny sposób co **loadGates**. Po odczytaniu wszystkich plików wejściowych, przechodzimy do funkcji **realize**, która to pobiera wszystkie odczytane dane, a następnie wykonuje operacje symulacji układu.Następnie rezultat jest wypisywany na ekran poprzez funkcje **print**. Na sam koniec rezultat wynikowy jest zapisywany w pliku zewnętrznym poprzez metodę **save**.

**4.2 Szczegółowy opis typów i funkcji**

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

**5 Testowanie**

Program został przetestowany na różnego rodzaju plikach. Pliki niepoprawne (nie zawierające oczekiwanego formatu, niezgodne ze specyfikacją) powodują zgłoszenie błędu oraz wypisanie stosownego komunikatu. Program został sprawdzony pod kątem wycieków pamięci.

**6 Wnioski**

Zrealizowałem zadanie, które polegało na napisaniu programu do symulacji układu bramek logicznych. Program jest prosty i przyjemny dla użytkownika. Pisząc program nauczyłem się obsługi kontenera STL – vector oraz wskaźników inteligentnych – shared\_ptr, które to bardzo usprawniają pisanie kodu.