Politechnika Śląska w Gliwicach

Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

TUC

autor	Dorian Barański
prowadzący	mgr inż. Dariusz Marek
rok akademicki	2018/2019
kierunek	Informatyka
rodzaj studiów	SSI
semestr	3
termin laboratorium / ćwiczeń	poniedziałek, 17:00 – 18:30
grupa	6
sekcja	19
termin oddania sprawozdania	2019-01-19
data oddania sprawozdania	2019-01-18

1. Wstęp

Celem projektu było napisanie programu symulującego działanie układu cyfrowego, złożonego z bramek logicznych. Dostępne były następujące bramki: AND, NAND, OR, NOR, XOR, XNOR oraz NEG. Każda bramka ma jedno wyjście i dwa wejścia za wyjątkiem bramki NEG, która ma tylko jedno wejście. Połączenie wejść i wyjść bramek jest traktowane jako węzeł.

Plik wejściowy przedstawiający układ na 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>

Przykładowo:
IN: 1 6
OUT: 3
NAND 1 6 5
NAND 1 5 2
NAND 5 6 4
NAND 2 4 3

Drugi plik wejściowy zawiera w każdej linii stany wejść, dla których należy znaleźć stan wyjść:
1: 0 6:0
1:0 6:1
1:1 6:0
1:1 6:1

Plik wynikowy podaje wartości wyjść dla zadanych stanów wejść.
IN: 1:0 6:0 OUT: 3:0
IN: 1:0 6:1 OUT: 3:1
IN: 1:1 6:0 OUT: 3:1
IN: 1:1 6:1 OUT: 3:0

2 Analiza zadania

Celem jest przeprowadzenie symulacji układu cyfrowego pod względem poprawności działania oraz uzyskania wartości węzła wyjściowego układu dla różnych wartości węzłów wejściowych.

2.1 Struktury danych

W programie wykorzystano listę dwukierunkową, której elementami są węzły. Węzły reprezentują dwa wejścia, wyjście oraz wartości jakie przyjmują.

2.2 Algorytmy

Dane wejściowe są wczytywane z plików .txt , następnie zostaje przeprowadzona symulacja układu, a uzyskane wartości wyjściowe są zapisywane w pliku wynikowym w formacie .txt . Ostatnim krokiem jest zwolnienie pamięci po przez usunięcie listy.

3. Specyfikacja zewnętrzna

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

-u plik wejściowy z układem
-i plik wejściowy ze stanami wejść
-o plik wyjściowy ze stanami wyjść

Przykładowe uruchomienie programu:

TUC.exe –i dane001.txt –u uklad001.txt –o wynik.txt

4. Typy

W programie zdefiniowano następujący typ:

```
struct wezel
{
  string  funkcja; //!funkcja logiczna
  int     we1; //!wejscie pierwsze
  int   we1val; //1 wartosc wejscia pierwszego
  int     we2; //!wejscie drugie
  int   we2val; //!wartosc wejscia drugiego
  int     wyf;//!wyjscie wezla
  int   wyfval;//!wartosc wyjscia
  wezel   *nst; //!wskaznik na nastepny element listy
  wezel   *pop; //!wskaznik na poprzedni element lsity
};
```

Struktura jest listą dwukierunkową, która przechowuje informacje o węźle układu.

4.1 Opis implementacji funkcji

W programie zdefiniowano następujące funkcje :

```
void OpcjeProgramu(void);
```

Powyższa funkcja wyświetla informacje na temat przykładowego uruchomienia. Jest to pomoc w przypadku ewentualnych problemów przy odpaleniu programu.

```
bool CzyDozwolony(string przelacznik);
```

Sprawdza czy użyte przez użytkownika przerzutniki maja poprawna formę.
Musza mieć długość równą 2, a ich kolejność może być dowolna.
W przypadku nieodpowiedniego formatu zostaje podana informacja o popełnionym bledzie.

parametry funkcji:

-i - Przerzutnik do pliku z danymi wejściowymi
-u - Przerzutnik do pliku z układem
-o - Przerzutnik do pliku wyjściowego

```
void DrukujUC(wezel *g);
```

Funkcja wywołuje funkcje konwersji typu int do string, następnie drukuje listę dwukierunkową w postaci:
we1:we1val we2:we2val wyf:wyfval
Przykładowo:
1:0 6:0 3:0

```
void UsunListeWezlow(wezel *&glowa);
```

Funkcja usuwająca listę w celu zwolnienia pamięci. Sprząta po sobie.

parametry funkcji:

wezel *&glowa - wskaźnik na pierwszy element listy

```
void TworzWezel(string funkcja, int we1, int we1val, int we2, int
we2val, int wyf, int wyfval);
```

Funkcja tworzy nowy węzeł i umieszcza w liście na miejscu zależnym od tego
ile elementów znajduje sie aktualnie w liście.

parametry funkcji:

wezel *NowyW - wskaźnik na nowo powstały element listy

```
void WczytajWezel(string ln);
```

Funkcja konstruuje listę dwukierunkową. Dla każdego IN: i OUT: tworzy węzły w zależności od tego ile maja wejść i wyjść.
W przypadku bramek dwuwejściowych występują dwa wejścia i jedno wyjście.
Natomiast wyjątek stanowi bramka NEG, która posiada tylko jedno wejście, drugie jest pomijane.
Początkowe wartości elementu maja przypisana wartość -1, czyli nieistniejąca wartość logiczna (pusta).
W przypadku błędnych danych zostaje podane informacje o bledzie.

parametry funkcji:

int ilParam - ilość parametrów
char *param - wskaźnik na parametr

```
void TUC(char* FNameUklad, char *FNameDane, char *FNameWynik);
```

Funkcja wczytuje uporządkowany plik z danymi.
Porządkuje plik z układem zastępując wielokrotne wystąpienia spacji i tabulacji pojedynczym znakiem białym. Inicjuje plik wyników (otwiera pusty plik). W przypadku gdy nie udało sie otworzyć pliku wejściowego lub wyjściowego, zostaje wygenerowana informacja o bledzie.

parametry funkcji:

char* FNameUklad - ścieżka dostępu i nazwa pliku ze schematem układu
char* FNameDane - ścieżka dostępu i nazwa pliku z danymi
char* FNameWynik - ścieżka dostępu i nazwa pliku wynikowego

```
void ZerowanieUC(void);
```

Funkcja czyści listę przed załadowaniem nowych danych do symulacji.
Czyli ustawia wartości wejść i wyjść elementów na -1 (Nieistniejąca wartość logiczna)

parametry funkcji:

wezel *g - wskaźnik na pierwszy element listy

```
void WczytajDANE(char *FNameDane);
```

Funkcja ma za zadanie wczytać plik z danymi i sprawdzić czy jego zawartość jest poprawna.
Gdy jest wszystko jest poprawne, wówczas następuje szereg operacji związanych z testowaniem układu. Jednak gdy plik jest niepoprawny, wtedy zostaje podana informacja o bledzie.

parametry funkcji:

char *FNameDane - ścieżka dostępu i nazwa pliku z danymi

```
void DANEdoUC(wezel *g, string ln);
```

Funkcja wprowadza dane do układu cyfrowego i przede wszystkim kontroluje ich ilość i format. Musza sie one zgadzać z wczytanym schematem w przeciwnym wypadku zostaną podane informacje o bledach.

parametry funkcji:

wezel *g - wskaźnik na element listy (węzeł)

```
bool WczytajWartWezlaIN(int Wezel, int wartosc);
```

Funkcja przypisuje wejściom i wyjściom odpowiednia wartość.
Zwraca wartość true gdy wszystkie węzły IN maja nadana wartość.
Wpisuje <Wartosc> <Wezla> wszystkich węzłów w których występuje.

parametry funkcji:

int wartosc - wartość która zostanie przypisana wejściom, wyjściom

---

```
int ObliczWyFValWezela(int nrWezla);
```

Jest to funkcja rekurencyjna, czyli wywołuje samą siebie.
Gdy wartości wejść wynoszą -1, najpierw zostaje im przypisana odpowiednia wartość.
Następnie zostaje obliczona wartość na wyjściu.

parametry funkcji:

wezel *glowa - wskaźnik na pierwszy element listy
wezel *ogon - wskaźnik na ostatni element listy

```
wezel *SzukWezel(int nrw, wezel *&w);
```

Przeszukuje listę w celu znalezienia odpowiedniego węzła.

parametry funkcji:

wezel *&w - wskaznik na element listy

```
void SymulacjaUC(wezel *g, wezel *o);
```

Funkcja przeprowadza symulacje układu cyfrowego w poszukiwaniu wartości węzła wyjściowego układu.
Przeszukuje cala listę do momentu znalezienia węzła wyjściowego, wówczas oblicza jego wartość.

parametry funkcji:

wezel *glowa - wskaźnik na pierwszy element listy
wezel *ogon - wskaźnik na ostatni element listy

```
void WynikiDoPliku(wezel *g, char *FNameWynik);
```

Funkcja przeszukuje listę w celu pozyskania informacji o numerach węzłów wejściowych, o numerze węzła wyjściowego oraz o wartościach jakie przyjmują.
Zapisuje otrzymane wyniki do pliku wynikowego w przykładowej postaci:
IN: 1:0 6:0 OUT: 3:0
IN: 1:0 6:1 OUT: 3:1
IN: 1:1 6:0 OUT: 3:1
IN: 1:1 6:1 OUT: 3:0

parametry funkcji:
wezel *g - wskaźnik na pierwszy element listy
char *FNameWynik - ścieżka dostępu i nazwa pliku wynikowego

```
int ileznk(string l, char znk);
```

Powyższa funkcja sprawdza ilość znaków w linii.

parametry funkcji:

string l - pojedyncza linia
char znk - pojedynczy znak

```
bool testlinidadanych(string l);
```

Funkcja sprawdza poprawność znaków występujących w linii.
Dozwolone są tylko liczby 0-9, : , znak biały oraz tabulacja.

parametry funkcji:

string l - pojedyncza linia

```
bool czytajWW(string &l, int &wart);
```

Funkcja usuwa ewentualne spacje wiodące. Dokonuje konwersji z łańcucha znaków do wartości całkowitych int.

parametry funkcji:

string &l - referencja do linii l

```
int FnLog(string fun, int w1v, int w2v);
```

Jest to funkcja logiczna, która oblicza wartości wyjściowe w zależności od wykorzystywanych bramek.
Bramki "AND", "NAND", "OR", "NOR", "XOR", "XNOR" są dwuwejściowego, dlatego do obliczenia wartości wyjściowej
potrzebne są wartości wejścia pierwszego i drugiego.
Natomiast bramka "NEG" wykorzystuje tylko jedno wejście, drugie jest bez znaczenia, czyli zostaje pominięte.
W przypadku nie rozpoznania bramki, funkcja logiczna przyjmuje wartość -1, czyli wartość 'nieistniejąca'

parametry funkcji:

int w1v - wartość pierwszego wejścia
int w2v - wartość drugiego wejścia

```
string its(int n);
```

Funkcja konwertuje typ integer do typu string

parametry funkcji:

string &l - referencja do linii l

5 Testowanie

Program został przetestowany na różnego rodzaju plikach.
Wielokrotne wystąpienia spacji lub tabulacji nie powodują żadnych negatywnych skutków. Informacje o błędzie może spowodować:

-niepoprawna liczba parametrów w linii
-niedozwolone operatory logiczne
-niepoprawny format danych oraz plików wejściowych

Kolejność używania przerzutników może być dowolna lecz muszą być użyte w sposób poprawny i zgodny z instrukcją.

6 Wnioski

Program TUC, którego zadaniem było przeprowadzenie symulacji układu cyfrowego jest programem trudnym zarówno od strony algorytmicznej jak i od strony dynamicznej. Wymagał wielu złożonych funkcji, a zatem obszernej wiedzy i umiejętnej pracy na listach dwukierunkowych oraz wskaźnikach. Czego wynikiem jest fakt, że nie byłem w stanie w pełni pozbyć się wycieków pamięci.
Pomimo tego, dzięki temu programowi zgłębiłem swoją wiedzę w części dynamicznej.

Literatura

- Jerzy Grębosz. Symfonia C++ standard. Wydawnictwo, EDITION 2000, Kraków, 2000.
- http://info.wsisiz.edu.pl/~sulewskp/doxygen/komentowanie.html?fbclid=IwAR18hvab76Y5Lwiy1tOYLNcZVahgyjvTMyZcYdSFSFFA9WAdOZ2RJzhOqc8