

Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 2

Symulacja

autor	Michał Ślusarczyk
prowadzący	mgr. Piotr Pecka
rok akademicki	2019/2020
kierunek	Informatyka
rodzaj studiów	SSI
semestr	2
termin laboratorium	środa, 15:30 - 17:00
sekcja	

termin oddania sprawozdania	2019-08-31
-----------------------------	------------

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, za wyjątkiem bramki neg, która ma jedno wejście i jedno wyjście. Połączenie wejść i wyjść bramek jest traktowane jako węzeł. Zmiana stanu na wyjściu bramki następuje po jednym kroku taktu. Na poniższym rysunku zaznaczono węzły prostego układu.

Plik wejściowy przedstawiający układ na następujący format: W pierwszej linii podane są numery węzłów będących wejściami układu.

W drugiej linii numery węzłów będące wyjściami układu.

Każda następna linia zawiera opis jednej bramki w postaci: Plik z układem dla powyższego rysunku ma postać:

IN: 1 4

OUT: 6 7

NEG 1 2

NAND 4 2 3

NAND 1 4 5

NAND 2 4 3

NAND 3 7 6

NAND 5 6 7

Drugi plik wejściowy zawiera w każdej linii stany wejść i czas ich trwania w taktach, dla których należy znaleźć stan wyjść, :

1:0 4:1 3

1:0 4:0 4

1:1 4:0 2

1:1 4:1 3

1:1 4:0 3

1:0 4:0 4

Plik wynikowy podaje wartości wyjść w kolejnych taktach dla zadanych stanów wejść. Wyjście bramki może mieć wartości 0, 1 lub X, gdy jeszcze stan nie został ustalony:

IN: 1:0 4:1 T:1 OUT: 6:x 7:x

IN: 1:0 4:1 T:2 OUT: 6:x 7:x

IN: 1:0 4:1 T:3 OUT: 6:1 7:x

IN: 1:0 4:0 T:4 OUT: 6:1 7:0

IN: 1:0 4:0 T:5 OUT: 6:1 7:0

IN: 1:0 4:0 T:6 OUT: 6:1 7:0

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna): -u plik wejściowy z układem -i plik wejściowy ze stanami wejść -o plik wyjściowy ze stanami wyjść

Analiza zadania

Zagadnienie przedstawia problem interpretacji układów elektronicznych za pomocą algorytmów przechowywania danych w optymalny sposób oraz operowaniu na nich.

2.1 Struktury danych

W programie wykorzystano pięć różnych struktur danych które są ze sobą ściśle powiązane. Pozwalają one na zapis danych w postaci list.

2.2 Algorytmy

Program nie wymaga algorytmów sortujących. Posiada on mechanizmy pozwalające symulować uproszczony w składowe układ elektroniczny składający się z bramek

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików: dwóch wejściowych oraz wyjściowego po odpowiednich przełącznikach -i, -u oraz -o, np:

program.exe -i Data1.txt, -u Data2.txt, -o Data3.txt

Pliki posiadają rozszerzenie tekstowe.

Uruchomienie programu z gdzie plik wejściowy jest pusty wyświetla się komunikat:

Plik pusty!

Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie odpowiedniego komunikatu:

Nie ma takiego pliku !

Podanie nieprawidłowych danych powoduje wyświetlenie odpowiedniego komunikatu:

Niepoprawny format danych!

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (realizacja układu).

4.1 Ogólna struktura programu

W funkcji głównej main po spełnieniu warunków związanych z poprawnym uruchomieniem programu w konsoli, wywoływana jest funkcja główna Management w której znajdują się wywołania metod odpowiedzialnych za realizację poszczególnych segmentów programu.

Po wczytaniu danych za pomocą funkcji Load1 i Load2 do odpowiednich struktur następuje wykorzystanie funkcji Attribution przygotowującej struktury do dalszych operacji.

Następnie za pomocą funkcji CompeteData, MakeSolution, SolveStep oraz CompleteStep realizowane jest rozwiązanie zagadnienia oraz przygotowanie rozwiązania do zapisu oraz wyświetlenia w konsoli.

Ostatnim krokiem jest zapis danych do pliku wynikowego oraz wyświetlenie rozwiązania w konsoli.

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. Plik pusty nie powoduje zgłoszenia błędu, ale utworzenie pustego pliku wynikowego (został podany pusty zbiór liczb i pusty zbiór został zwrócony). Przy braku pliku wejściowego o podanej nazwie zwracany jest odpowiedni komunikat. Niepoprawny format danych powoduje przerwanie programu oraz wyświetlenie odpowiedniego komunikatu. Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program realizujący układ elektroniczny składający się z bramek wymagał wiedzy z zakresu zarządzania pamięcią w sposób dynamiczny, umiejętności zapisu i odczytu z pliku, radzenia sobie z zagnieżdżoną strukturą danych oraz sprawdzał świadomość piszącego na temat jego programu. Dodatkowo należy posiadać wiedzę z zakresu teorii układów cyfrowych aby zrealizować zagadnienie w sposób zgodny z rzeczywistością. Najtrudniejszym elementem w mojej opinii było zinterpretowanie polecenia w sposób umożliwiający jego realizację za pomocą kodu.

Osobiście uważam że sam projekt był wielce pouczający i pozwolił w sposób kreatywny wykorzystać dotychczas zdobytą wiedzę oraz uzupełnić ewentualne braki.

Literatura

Jerzy Grębosz "Symfonia C++ standard", Tom 1 i
2

¹<http://sjp.pwn.pl>

²<http://sjp.pwn.pl/zasady>

³[https://pl.wikipedia.org/wiki/Pomoc:Powszechne błędy językowe](https://pl.wikipedia.org/wiki/Pomoc:Powszechne_błędy_językowe)

Dodatek

Szczegółowy opis typów i funkcji

Symulacja

Wygenerowano przez Doxygen 1.8.20

1 Indeks klas	1
1.1 Lista klas	1
2 Indeks plików	3
2.1 Lista plików	3
3 Dokumentacja klas	5
3.1 Dokumentacja struktury Input	5
3.1.1 Opis szczegółowy	5
3.2 Dokumentacja struktury LInt	5
3.2.1 Opis szczegółowy	6
3.3 Dokumentacja struktury Node	6
3.3.1 Opis szczegółowy	6
3.4 Dokumentacja struktury Solution	6
3.4.1 Opis szczegółowy	7
3.5 Dokumentacja struktury System	7
3.5.1 Opis szczegółowy	7
4 Dokumentacja plików	9
4.1 Dokumentacja pliku Functions.c	9
4.1.1 Opis szczegółowy	9
4.1.2 Dokumentacja funkcji	9
4.1.2.1 Attribution()	10
4.1.2.2 CompleteData()	10
4.1.2.3 CompleteStep()	10
4.1.2.4 Load1()	11
4.1.2.5 Load2()	11
4.1.2.6 MakeSolution()	12
4.1.2.7 Management()	12
4.1.2.8 Save()	12
4.1.2.9 ShowAll()	13
4.1.2.10 SolveStep()	13
4.2 Dokumentacja pliku Functions.h	14
4.2.1 Opis szczegółowy	14
4.2.2 Dokumentacja funkcji	14
4.2.2.1 Attribution()	14
4.2.2.2 CompleteData()	15
4.2.2.3 CompleteStep()	15
4.2.2.4 Load1()	15
4.2.2.5 Load2()	16
4.2.2.6 MakeSolution()	16
4.2.2.7 Management()	17
4.2.2.8 Save()	17

4.2.2.9 ShowAll()	18
4.2.2.10 SolveStep()	18
4.3 Dokumentacja pliku Lists.c	18
4.3.1 Opis szczegółowy	19
4.3.2 Dokumentacja funkcji	19
4.3.2.1 AddInputEnd()	19
4.3.2.2 AddNodeEnd()	19
4.3.2.3 AddSolutionEnd()	20
4.3.2.4 AddSystemEnd()	20
4.3.2.5 CompareNode()	20
4.3.2.6 CopyLIntEnd()	20
4.3.2.7 CopyNodeEnd()	20
4.3.2.8 CopyNodeList()	20
4.3.2.9 CreateInput()	21
4.3.2.10 CreateLInt()	21
4.3.2.11 CreateNode()	21
4.3.2.12 CreateSolution()	21
4.3.2.13 CreateSystem()	21
4.3.2.14 FindLGate()	21
4.3.2.15 FindNodeID()	21
4.3.2.16 FindNodeN()	22
4.3.2.17 FindNodeVAL()	22
4.3.2.18 PrintNode()	22
4.3.2.19 ShowInput()	22
4.3.2.20 ShowNode()	22
4.3.2.21 ShowSolution()	22
4.3.2.22 ShowSystem()	23
4.4 Dokumentacja pliku Lists.h	23
4.4.1 Opis szczegółowy	23
4.4.2 Dokumentacja funkcji	23
4.4.2.1 AddInputEnd()	24
4.4.2.2 AddNodeEnd()	24
4.4.2.3 AddSolutionEnd()	24
4.4.2.4 AddSystemEnd()	24
4.4.2.5 CompareNode()	24
4.4.2.6 CopyLIntEnd()	24
4.4.2.7 CopyNodeEnd()	25
4.4.2.8 CopyNodeList()	25
4.4.2.9 CreateInput()	25
4.4.2.10 CreateLInt()	25
4.4.2.11 CreateNode()	25
4.4.2.12 CreateSolution()	25

4.4.2.13 CreateSystem()	25
4.4.2.14 FindLGate()	26
4.4.2.15 FindNodeID()	26
4.4.2.16 FindNodeN()	26
4.4.2.17 FindNodeVAL()	26
4.4.2.18 PrintNode()	26
4.4.2.19 ShowInput()	26
4.4.2.20 ShowNode()	27
4.4.2.21 ShowSolution()	27
4.4.2.22 ShowSystem()	27
4.5 Dokumentacja pliku Source.c	27
4.5.1 Opis szczegółowy	27
4.6 Dokumentacja pliku Structures.h	27
4.6.1 Opis szczegółowy	28
4.6.2 Dokumentacja definicji typów	28
4.6.2.1 bool	28
4.6.2.2 Input	28
4.6.2.3 LInt	29
4.6.2.4 Node	29
4.6.2.5 Solution	29
4.6.2.6 System	29
4.6.2.7 Type	29
4.6.3 Dokumentacja typów wyliczanych	29
4.6.3.1 bool	29
4.6.3.2 Type	29
Indeks	31

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Input	5
LInt	5
Node	6
Solution	6
System	7

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Functions.c	9
Functions.h	14
Lists.c	18
Lists.h	23
Source.c	27
Structures.h	27

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury Input

```
#include <Structures.h>
```

Atrybuty publiczne

- struct [Node](#) * [i](#)
Lista wezlow wejscia.
- int [tact](#)
- struct [Input](#) * [pNext](#)
Wskaźnik na kolejny element listy.

3.1.1 Opis szczegółowy

Struktura wejscia

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Structures.h](#)

3.2 Dokumentacja struktury LInt

```
#include <Structures.h>
```

Atrybuty publiczne

- int [val](#)
Wartosc int elementu listy.
- struct [LInt](#) * [pNext](#)
Wskaźnik na kolejny element listy.

3.2.1 Opis szczegółowy

Struktura listy intow

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Structures.h](#)

3.3 Dokumentacja struktury Node

```
#include <Structures.h>
```

Atrybuty publiczne

- char [val](#)
Atrybut odpowiadający możliwemu stanowi wezła.
- unsigned int [id](#)
Atrybut określający id wezła.
- struct [Node](#) * [pNext](#)
Wskaźnik na kolejny element listy.

3.3.1 Opis szczegółowy

Struktura wezła

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Structures.h](#)

3.4 Dokumentacja struktury Solution

```
#include <Structures.h>
```

Atrybuty publiczne

- struct [Node](#) * [tempST](#)
Wskaźnik do listy tymczasowej podczas określania rozwiązania.
- struct [LInt](#) * [lgate](#)
Wskaźnik do listy tymczasowej podczas określania rozwiązania.
- struct [Node](#) * [ST](#)
Wskaźnik do listy rozwiązań.
- int [tact](#)
Atrybut określający liczbę taktów w danym rozwiązaniu.
- struct [Solution](#) * [pNext](#)
Wskaźnik na kolejny element listy.

3.4.1 Opis szczegółowy

Struktura rozwiązan

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Structures.h](#)

3.5 Dokumentacja struktury System

```
#include <Structures.h>
```

Atrybuty publiczne

- unsigned int [gid](#)
A trybut określający id bramki.
- enum type [gate](#)
Atrybut określający typ bramki.
- struct [Node](#) * [in](#)
Wskaźnik do listy wejść bramki.
- struct [Node](#) * [out](#)
Wskaźnik do listy wyjść bramki.
- struct [System](#) * [pNext](#)
Wskaźnik na kolejny element listy.

3.5.1 Opis szczegółowy

Struktura bramki

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Structures.h](#)

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku Functions.c

```
#include "Functions.h"
```

Funkcje

- `bool Management` (const char *path1, const char *path2, const char *path3)
- `bool Load1` (const char *path1, struct `System` *`ϕ_S`, struct `Node` *`ϕ_CI`, struct `Node` *`ϕ_O`, struct `Node` *`ϕ_ST`)
- `bool Load2` (const char *path2, struct `Node` *`ϕ_CI`, struct `Input` *`ϕ_I`, struct `Node` *`ϕ_ST`)
- void `Attribution` (FILE *file, `Node` *`ϕ_ST`, `Node` *`ϕ_O`, `System` *`ϕ_S`, `Node` *`ϕ_CI`, `Type` type_name, int i)
- `bool CompleteData` (`Input` *`ϕ_I`, `Node` *`ϕ_ST`, `Solution` *`ϕ_SOL`)
- void `MakeSolution` (`System` *`ϕ_S`, `Solution` *`ϕ_SOL`)
- void `SolveStep` (`System` *tempS, `Solution` *`ϕ_tempLST`)
- void `CompleteStep` (`Node` *tempST, `Node` *`ϕHeadST`)
- void `ShowAll` (struct `System` *`ϕHeadS`, struct `Node` *`ϕHeadST`, struct `Node` *`ϕHeadCI`, struct `Input` *`ϕHeadI`, struct `Node` *`ϕHeadO`, struct `Solution` *`ϕHeadSOL`)
- `bool Save` (const char *path3, struct `Input` *`ϕHeadI`, struct `Node` *`ϕHeadO`, struct `Solution` *`ϕHeadSOL`)

4.1.1 Opis szczegółowy

4.1.2 Dokumentacja funkcji

4.1.2.1 Attribution()

```
void Attribution (
    FILE * file,
    Node ** d_ST,
    Node ** d_O,
    System ** d_S,
    Node ** d_CI,
    Type type_name,
    int i )
```

Funkcje do odpowiedniego przypisania wczytanych danych

Parametry

<i>path2</i>	Druga ścieżka wejściowa
<i>file</i>	Zmienna odpowiedzialna za strumień do pliku
<i>d_ST</i>	Podwojny wskaźnik do listy stanu
<i>d_O</i>	Podwojny wskaźnik do listy wyjsc
<i>d_S</i>	Podwojny wskaźnik do listy bramek
<i>d_CI</i>	Podwojny wskaźnik do listy wejść z pierwszego pliku
<i>i</i>	Odczytane z pliku wejściowego id bramki

4.1.2.2 CompleteData()

```
bool CompleteData (
    Input * d_I,
    Node ** d_ST,
    Solution ** d_SOL )
```

Funkcja do odpowiedniego przygotowania danych do rozwiązania problemu

Parametry

<i>d_I</i>	Podwojny wskaźnik do listy wyjściowej
<i>d_ST</i>	Podwojny wskaźnik do listy stanu
<i>d_SOL</i>	Podwojny wskaźnik do listy rozwiązań

4.1.2.3 CompleteStep()

```
void CompleteStep (
    Node * tempST,
    Node ** pHeadST )
```

Funkcja pomocnicza do przypisania rozwiązania problemu

Parametry

<i>tempST</i>	Wskaźnik do listy rozwiązań fragmentu problemu
<i>pHeadST</i>	Wskaźnik do listy stanu do którego ma być przypisany tempST

4.1.2.4 Load1()

```
bool Load1 (
    const char * path1,
    struct System ** d_S,
    struct Node ** d_CI,
    struct Node ** d_O,
    struct Node ** d_ST )
```

Funkcja do odczytu danych

Parametry

<i>path1</i>	Pierwsza ścieżka wejściowa
<i>d_S</i>	Podwojny wskaźnik do listy bramek
<i>d_CI</i>	Podwojny wskaźnik do listy wejść z pierwszego pliku
<i>d_O</i>	Podwojny wskaźnik do listy wyjść
<i>d_ST</i>	Podwojny wskaźnik do listy stanu

Zwraca

Zwraca poprawność wykonania funkcji

4.1.2.5 Load2()

```
bool Load2 (
    const char * path2,
    struct Node ** d_CI,
    struct Input ** d_I,
    struct Node ** d_ST )
```

Funkcja do odczytu danych

Parametry

<i>path2</i>	Druga ścieżka wejściowa
<i>d_CI</i>	Podwojny wskaźnik do listy wejść z pierwszego pliku
<i>d_I</i>	Podwojny wskaźnik do listy wyjściowej
<i>d_ST</i>	Podwojny wskaźnik do listy stanu

Zwraca

Zwraca poprawnosc wykonania funkcji

4.1.2.6 MakeSolution()

```
void MakeSolution (
    System ** d_S,
    Solution ** d_SOL )
```

Funkcja sluzaca do rozwiazania problemu

Parametry

<i>d_S</i>	Podwojny wskaznik do listy bramek
<i>d_SOL</i>	Podwojny wskaznik do listy rozwiazan

4.1.2.7 Management()

```
bool Management (
    const char * path1,
    const char * path2,
    const char * path3 )
```

Główna funkcja w której wywoływane są funkcje składowe

Parametry

<i>path1</i>	Pierwsza sciezka wejscowa
<i>path2</i>	Druga sciezka wejscowa
<i>path3</i>	Sciezka pliku wyjsciowego

Zwraca

Zwraca poprawnosc wykonania funkcji

4.1.2.8 Save()

```
bool Save (
    const char * path3,
    struct Input * pHeadI,
    struct Node * pHeadO,
    struct Solution * pHeadSOL )
```

Funkcja do zapisu rozwiazania do pliku

Parametry

<i>path3</i>	Sciezka wyjsciowa
<i>pHeadI</i>	Wskaznik do listy wejsc
<i>pHeadO</i>	Wskaznik do listy wyjsc
<i>pHeadSOL</i>	Wskaznik do listy rowiazan

Zwraca

Zwraca poprawnosc wykonania funkcji

4.1.2.9 ShowAll()

```
void ShowAll (
    struct System * pHeadS,
    struct Node * pHeadState,
    struct Node * pHeadCI,
    struct Input * pHeadI,
    struct Node * pHeadO,
    struct Solution * pHeadSOL )
```

Funkcja do wyswietlania rozwiazania

Parametry

<i>pHeadS</i>	Wskaznik do listy vramek
<i>pHeadState</i>	Wskaznik do listy stanow
<i>pHeadCI</i>	Wskaznik do listy wejsc z pierwszego pliku
<i>pHeadI</i>	Wskaznik do listy wejsc
<i>pHeadO</i>	Wskaznik do listy wyjsc
<i>pHeadSOL</i>	Wskaznik do listy rowiazan

4.1.2.10 SolveStep()

```
void SolveStep (
    System * tempS,
    Solution ** d_tempLST )
```

Funkcja pomocnicza do rowiazania fragmentu problemu

Parametry

<i>tempS</i>	Lista bramek
<i>d_tempLST</i>	Tymczasowa lista rozwiazan fragmentu problemu

4.2 Dokumentacja pliku Functions.h

```
#include "structures.h"
#include "Lists.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
```

Funkcje

- **bool Management** (const char *path1, const char *path2, const char *path3)
- **bool Load1** (const char *path1, struct **System** * d_S, struct **Node** * d_CI, struct **Node** * d_O, struct **Node** * d_ST)
- **bool Load2** (const char *path2, struct **Node** * d_CI, struct **Input** * d_I, struct **Node** * d_ST)
- **bool Save** (const char *path3, struct **Input** *pHeadI, struct **Node** *pHeadO, struct **Solution** *pHeadSOL)
- void **Attribution** (FILE *file, **Node** * d_ST, **Node** * d_O, **System** * d_S, **Node** * d_CI, **Type** type_name, int i)
- **bool CompleteData** (**Input** *d_I, **Node** * d_ST, **Solution** * d_SOL)
- void **MakeSolution** (**System** * d_S, **Solution** * d_SOL)
- void **SolveStep** (**System** *tempS, **Solution** * d_tempLST)
- void **CompleteStep** (**Node** *tempST, **Node** * pHeadST)
- void **ShowAll** (struct **System** *pHeadS, struct **Node** *pHeadState, struct **Node** *pHeadCI, struct **Input** *pHeadI, struct **Node** *pHeadO, struct **Solution** *pHeadSOL)

4.2.1 Opis szczegółowy

4.2.2 Dokumentacja funkcji

4.2.2.1 Attribution()

```
void Attribution (
    FILE * file,
    Node ** d_ST,
    Node ** d_O,
    System ** d_S,
    Node ** d_CI,
    Type type_name,
    int i )
```

Funkcje do odpowiedniego przypisania wczytanych danych

Parametry

<i>path2</i>	Druga sciezka wejscowa
<i>file</i>	Zmienna odpowiedzialna za strumien do pliku
<i>d_ST</i>	Podwojny wskaznik do listy stanu
<i>d_O</i>	Podwojny wskaznik do listy wyjsc
<i>d_S</i>	Podwojny wskaznik do listy bramek
<i>d_Cl</i>	Podwojny wskaznik do listy wejsc z pierwszego pliku
<i>i</i>	Odczytane z pliku wejscowego id bramki

4.2.2.2 CompleteData()

```
bool CompleteData (
    Input * d_I,
    Node ** d_ST,
    Solution ** d_SOL )
```

Funkcja do odpowiedniego przygotowania danych do rowiazania problemu

Parametry

<i>d_I</i>	Podwojny wskaznik do listy wyjscowej
<i>d_ST</i>	Podwojny wskaznik do listy stanu
<i>d_SOL</i>	Podwojny wskaznik do listy rozwiazan

4.2.2.3 CompleteStep()

```
void CompleteStep (
    Node * tempST,
    Node ** pHeadST )
```

Funkcja pomocnicza do przypisania rozwiazania problemu

Parametry

<i>tempST</i>	Wskaznik do listy rozwiazan fragmentu problemu
<i>pHeadST</i>	Wskaznik do listy stanu do ktorego ma byc przypisany tempST

4.2.2.4 Load1()

```
bool Load1 (
    const char * path1,
```

```

struct System ** d_S,
struct Node ** d_CI,
struct Node ** d_O,
struct Node ** d_ST )

```

Funkcja do odczytu danych

Parametry

<i>path1</i>	Pierwsza sciezka wejscowa
<i>d_S</i>	Podwojny wskaznik do listy bramek
<i>d_CI</i>	Podwojny wskaznik do listy wejsc z pierwszego pliku
<i>d_O</i>	Podwojny wskaznik do listy wyjsc
<i>d_ST</i>	Podwojny wskaznik do listy stanu

Zwraca

Zwraca poprawnosc wykonania funkcji

4.2.2.5 Load2()

```

bool Load2 (
    const char * path2,
    struct Node ** d_CI,
    struct Input ** d_I,
    struct Node ** d_ST )

```

Funkcja do odczytu danych

Parametry

<i>path2</i>	Druga sciezka wejscowa
<i>d_CI</i>	Podwojny wskaznik do listy wejsc z pierwszego pliku
<i>d_I</i>	Podwojny wskaznik do listy wyjscowej
<i>d_ST</i>	Podwojny wskaznik do listy stanu

Zwraca

Zwraca poprawnosc wykonania funkcji

4.2.2.6 MakeSolution()

```

void MakeSolution (
    System ** d_S,
    Solution ** d_SOL )

```

Funkcja sluzaca do rozwiazania problemu

Parametry

<i>d_S</i>	Podwojny wskaźnik do listy bramek
<i>d_SOL</i>	Podwojny wskaźnik do listy rozwiązań

4.2.2.7 Management()

```
bool Management (
    const char * path1,
    const char * path2,
    const char * path3 )
```

Główna funkcja w której wywoływane są funkcje składowe

Parametry

<i>path1</i>	Pierwsza ścieżka wejściowa
<i>path2</i>	Druga ścieżka wejściowa
<i>path3</i>	Ścieżka pliku wyjściowego

Zwraca

Zwraca poprawność wykonania funkcji

4.2.2.8 Save()

```
bool Save (
    const char * path3,
    struct Input * pHeadI,
    struct Node * pHeadO,
    struct Solution * pHeadSOL )
```

Funkcja do zapisu rozwiązania do pliku

Parametry

<i>path3</i>	Ścieżka wyjściowa
<i>pHeadI</i>	Wskaźnik do listy wejść
<i>pHeadO</i>	Wskaźnik do listy wyjść
<i>pHeadSOL</i>	Wskaźnik do listy rozwiązań

Zwraca

Zwraca poprawność wykonania funkcji

4.2.2.9 ShowAll()

```
void ShowAll (
    struct System * pHeadS,
    struct Node * pHeadState,
    struct Node * pHeadCI,
    struct Input * pHeadI,
    struct Node * pHeadO,
    struct Solution * pHeadSOL )
```

Funkcja do wyświetlania rozwiązania

Parametry

<i>pHeadS</i>	Wskaźnik do listy vramek
<i>pHeadState</i>	Wskaźnik do listy stanów
<i>pHeadCI</i>	Wskaźnik do listy wejść z pierwszego pliku
<i>pHeadI</i>	Wskaźnik do listy wejść
<i>pHeadO</i>	Wskaźnik do listy wyjść
<i>pHeadSOL</i>	Wskaźnik do listy rozwiązań

4.2.2.10 SolveStep()

```
void SolveStep (
    System * tempS,
    Solution ** d_tempLST )
```

Funkcja pomocnicza do rozwiązania fragmentu problemu

Parametry

<i>tempS</i>	Lista ramek
<i>d_tempLST</i>	Tymczasowa lista rozwiązań fragmentu problemu

4.3 Dokumentacja pliku Lists.c

```
#include "Lists.h"
```

Funkcje

- `Node *CreateNode ()`

- `Input * CreateInput ()`
- `System * CreateSystem ()`
- `LInt * CreateLInt ()`
- `Solution * CreateSolution ()`
- `void AddNodeEnd (Node *n, Node * pHeadN)`
- `void CopyNodeEnd (Node *n, Node * pHeadN)`
- `void CopyNodeList (Node *pHeadN1, Node * pHeadN2)`
- `char FindNodeVAL (unsigned int tempid, Node *pHeadN)`
- `void ShowNode (Node *pHeadN)`
- `bool FindNodeID (unsigned int id, Node *pHeadState)`
- `void PrintNode (Node *n, Node *pHeadN)`
- `Node * FindNodeN (Node *n, Node * pHeadState)`
- `bool CompareNode (Node *pHeadI, Node *pHeadCI)`
- `void AddSystemEnd (System *s, System * pHeadS)`
- `void ShowSystem (System *pHeadS)`
- `void AddInputEnd (Input *i, Input * pHeadI)`
- `void ShowInput (Input *pHeadI)`
- `void CopyLIntEnd (unsigned int n, LInt * pHeadLInt)`
- `bool FindLGate (unsigned int gid, LInt *pHeadLInt)`
- `void AddSolutionEnd (Solution *sol, Solution * pHeadSOL)`
- `void ShowSolution (Input *pHeadI, Node *pHeadO, Solution *pHeadSOL)`

4.3.1 Opis szczegółowy

4.3.2 Dokumentacja funkcji

4.3.2.1 AddInputEnd()

```
void AddInputEnd (
    Input * i,
    Input ** pHeadI )
```

Funkcja dodająca wejście na koniec

4.3.2.2 AddNodeEnd()

```
void AddNodeEnd (
    Node * n,
    Node ** pHeadN )
```

Funkcja dodająca węzeł na koniec

4.3.2.3 AddSolutionEnd()

```
void AddSolutionEnd (
    Solution * sol,
    Solution ** pHeadSOL )
```

Funkcja kopiujaca wejscie na koniec

4.3.2.4 AddSystemEnd()

```
void AddSystemEnd (
    System * s,
    System ** pHeadS )
```

Funkcja dodajaca bramke na koniec

4.3.2.5 CompareNode()

```
bool CompareNode (
    Node * pHeadI,
    Node * pHeadCI )
```

Funkcja porownujaca wezly

4.3.2.6 CopyLIntEnd()

```
void CopyLIntEnd (
    unsigned int n,
    LInt ** pHeadLInt )
```

Funkcja kopiujaca element [LInt](#) na koniec listy

4.3.2.7 CopyNodeEnd()

```
void CopyNodeEnd (
    Node * n,
    Node ** pHeadN )
```

Funkcja kopiujaca wezel na koniec

4.3.2.8 CopyNodeList()

```
void CopyNodeList (
    Node * pHeadN1,
    Node ** pHeadN2 )
```

Funkcja kopiujaca liste wezlow

4.3.2.9 CreateInput()

```
Input* CreateInput ( )
```

Funkcja do tworzenia domyslniej struktury wejscia

4.3.2.10 CreateLInt()

```
LInt* CreateLInt ( )
```

Funkcja do tworzenia domyslniej struktury pomocniczej do listy typu Int

4.3.2.11 CreateNode()

```
Node* CreateNode ( )
```

Funkcja do tworzenia domyslniej struktury wezla

4.3.2.12 CreateSolution()

```
Solution* CreateSolution ( )
```

Funkcja do tworzenia domyslniej struktury rozwiazania

4.3.2.13 CreateSystem()

```
System* CreateSystem ( )
```

Funkcja do tworzenia domyslniej struktury bramki

4.3.2.14 FindLGate()

```
bool FindLGate (
    unsigned int gid,
    LInt * pHeadLInt )
```

Funkcja wyswietlajaca liste [LInt](#)

4.3.2.15 FindNodeID()

```
bool FindNodeID (
    unsigned int id,
    Node * pHeadN )
```

Funkcja znajdujaca wezel po ID

4.3.2.16 FindNodeN()

```
Node* FindNodeN (
    Node * n,
    Node ** pHeadState )
```

Funkcja znajdujaca konkretny wezel w liscie

4.3.2.17 FindNodeVAL()

```
char FindNodeVAL (
    unsigned int tempid,
    Node * pHeadN )
```

Funkcja Funkcja znajdujaca konkretny wezel w liscie, zwracajaca wartosc wezla

4.3.2.18 PrintNode()

```
void PrintNode (
    Node * n,
    Node * pHeadN )
```

Funkcja Funkcja znajdujaca konkretny wezel w liscie, i wyswietlajaca jego dane

4.3.2.19 ShowInput()

```
void ShowInput (
    Input * pHeadI )
```

Funkcja wyswietlajaca liste wejsc

4.3.2.20 ShowNode()

```
void ShowNode (
    Node * pHeadN )
```

Funkcja wyswietlajaca wezel

4.3.2.21 ShowSolution()

```
void ShowSolution (
    Input * pHeadI,
    Node * pHeadO,
    Solution * pHeadSOL )
```

Funkcja wyswietlajaca liste rozwiazan

4.3.2.22 ShowSystem()

```
void ShowSystem (
    System * pHeadS )
```

Funkcja wyswietlajaca liste bramek

4.4 Dokumentacja pliku Lists.h

```
#include "Structures.h"
#include "Functions.h"
#include <stdio.h>
#include <string.h>
```

Funkcje

- Node * CreateNode ()
- Input * CreateInput ()
- System * CreateSystem ()
- LInt * CreateLInt ()
- Solution * CreateSolution ()
- void AddNodeEnd (Node *n, Node * pHeadN)
- void CopyNodeEnd (Node *n, Node * pHeadN)
- void CopyNodeList (Node *pHeadN1, Node * pHeadN2)
- void ShowNode (Node *pHeadN)
- bool CompareNode (Node *pHeadI, Node *pHeadCI)
- bool FindNodeID (unsigned int id, Node *pHeadN)
- Node * FindNodeN (Node *n, Node * pHeadState)
- char FindNodeVAL (unsigned int tempid, Node *pHeadN)
- void PrintNode (Node *n, Node *pHeadN)
- void AddSystemEnd (System *s, System * pHeadS)
- void ShowSystem (System *pHeadS)
- void AddInputEnd (Input *i, Input * pHeadI)
- void ShowInput (Input *pHeadI)
- void CopyLIntEnd (unsigned int n, LInt * pHeadLInt)
- bool FindLGate (unsigned int gid, LInt *pHeadLInt)
- void AddSolutionEnd (Solution *sol, Solution * pHeadSOL)
- void ShowSolution (Input *pHeadI, Node *pHeadO, Solution *pHeadSOL)

4.4.1 Opis szczegółowy

4.4.2 Dokumentacja funkcji

4.4.2.1 AddInputEnd()

```
void AddInputEnd (
    Input * i,
    Input ** pHeadI )
```

Funkcja dodająca wejście na koniec

4.4.2.2 AddNodeEnd()

```
void AddNodeEnd (
    Node * n,
    Node ** pHeadN )
```

Funkcja dodająca węzeł na koniec

4.4.2.3 AddSolutionEnd()

```
void AddSolutionEnd (
    Solution * sol,
    Solution ** pHeadSOL )
```

Funkcja kopiująca wejście na koniec

4.4.2.4 AddSystemEnd()

```
void AddSystemEnd (
    System * s,
    System ** pHeadS )
```

Funkcja dodająca bramkę na koniec

4.4.2.5 CompareNode()

```
bool CompareNode (
    Node * pHeadI,
    Node * pHeadCI )
```

Funkcja porównująca węzły

4.4.2.6 CopyLIntEnd()

```
void CopyLIntEnd (
    unsigned int n,
    LInt ** pHeadLInt )
```

Funkcja kopiująca element [LInt](#) na koniec listy

4.4.2.7 CopyNodeEnd()

```
void CopyNodeEnd (
    Node * n,
    Node ** pHeadN )
```

Funkcja kopiujaca wezel na koniec

4.4.2.8 CopyNodeList()

```
void CopyNodeList (
    Node * pHeadN1,
    Node ** pHeadN2 )
```

Funkcja kopiujaca liste wezlow

4.4.2.9 CreateInput()

```
Input* CreateInput ( )
```

Funkcja do tworzenia domyslnej struktury wejscia

4.4.2.10 CreateLInt()

```
LInt* CreateLInt ( )
```

Funkcja do tworzenia domyslnej struktury pomocniczej do listy typu Int

4.4.2.11 CreateNode()

```
Node* CreateNode ( )
```

Funkcja do tworzenia domyslnej struktury wezla

4.4.2.12 CreateSolution()

```
Solution* CreateSolution ( )
```

Funkcja do tworzenia domyslnej struktury rozwiazania

4.4.2.13 CreateSystem()

```
System* CreateSystem ( )
```

Funkcja do tworzenia domyslnej struktury bramki

4.4.2.14 FindLGate()

```
bool FindLGate (
    unsigned int gid,
    LInt * pHeadLInt )
```

Funkcja wyswietlajaca liste [LInt](#)

4.4.2.15 FindNodeID()

```
bool FindNodeID (
    unsigned int id,
    Node * pHeadN )
```

Funkcja znajdujaca wezel po ID

4.4.2.16 FindNodeN()

```
Node* FindNodeN (
    Node * n,
    Node ** pHeadState )
```

Funkcja znajdujaca konkretny wezel w liscie

4.4.2.17 FindNodeVAL()

```
char FindNodeVAL (
    unsigned int tempid,
    Node * pHeadN )
```

Funkcja Funkcja znajdujaca konkretny wezel w liscie, zwracajaca wartosc wezla

4.4.2.18 PrintNode()

```
void PrintNode (
    Node * n,
    Node * pHeadN )
```

Funkcja Funkcja znajdujaca konkretny wezel w liscie, i wyswietlajaca jego dane

4.4.2.19 ShowInput()

```
void ShowInput (
    Input * pHeadI )
```

Funkcja wyswietlajaca liste wejsc

4.4.2.20 ShowNode()

```
void ShowNode (
    Node * pHeadN )
```

Funkcja wyswietlajaca wezel

4.4.2.21 ShowSolution()

```
void ShowSolution (
    Input * pHeadI,
    Node * pHeadO,
    Solution * pHeadSOL )
```

Funkcja wyswietlajaca liste rozwiazan

4.4.2.22 ShowSystem()

```
void ShowSystem (
    System * pHeadS )
```

Funkcja wyswietlajaca liste bramek

4.5 Dokumentacja pliku Source.c

```
#include <stdio.h>
#include "Functions.h"
```

Funkcje

- int **main** (int argc, char const *argv[])

4.5.1 Opis szczegółowy

4.6 Dokumentacja pliku Structures.h

```
#include <stdio.h>
#include <math.h>
```

Komponenty

- struct [Node](#)
- struct [System](#)
- struct [Input](#)
- struct [LInt](#)
- struct [Solution](#)

Definicje typów

- typedef enum [Type](#) [Type](#)
- typedef enum [bool](#) [bool](#)
- typedef struct [Node](#) [Node](#)
- typedef struct [System](#) [System](#)
- typedef struct [Input](#) [Input](#)
- typedef struct [LInt](#) [LInt](#)
- typedef struct [Solution](#) [Solution](#)

Wyliczenia

- enum [Type](#) {
 IN, OUT, NEG, AND,
 NAND, OR, NOR, XOR,
 XNOR, NA }
- enum [bool](#) { false, true }

4.6.1 Opis szczegółowy

4.6.2 Dokumentacja definicji typów

4.6.2.1 bool

```
typedef enum bool bool
```

Zdefiniowany typ bool'a

4.6.2.2 Input

```
typedef struct Input Input
```

Struktura wejścia

4.6.2.3 LInt

```
typedef struct LInt LInt
```

Struktura listy intow

4.6.2.4 Node

```
typedef struct Node Node
```

Struktura wezla

4.6.2.5 Solution

```
typedef struct Solution Solution
```

Struktura rozwiazan

4.6.2.6 System

```
typedef struct System System
```

Struktura bramki

4.6.2.7 Type

```
typedef enum Type Type
```

Zdefiniowany typ własny odpowiadający możliwym typom bramki

4.6.3 Dokumentacja typów wyliczanych

4.6.3.1 bool

```
enum bool
```

Zdefiniowany typ bool'a

4.6.3.2 Type

```
enum Type
```

Zdefiniowany typ własny odpowiadający możliwym typom bramki

Indeks

AddInputEnd
 Lists.c, [19](#)
 Lists.h, [23](#)

AddNodeEnd
 Lists.c, [19](#)
 Lists.h, [24](#)

AddSolutionEnd
 Lists.c, [19](#)
 Lists.h, [24](#)

AddSystemEnd
 Lists.c, [20](#)
 Lists.h, [24](#)

Attribution
 Functions.c, [9](#)
 Functions.h, [14](#)

bool
 Structures.h, [28](#), [29](#)

CompareNode
 Lists.c, [20](#)
 Lists.h, [24](#)

CompleteData
 Functions.c, [10](#)
 Functions.h, [15](#)

CompleteStep
 Functions.c, [10](#)
 Functions.h, [15](#)

CopyLIntEnd
 Lists.c, [20](#)
 Lists.h, [24](#)

CopyNodeEnd
 Lists.c, [20](#)
 Lists.h, [24](#)

CopyNodeList
 Lists.c, [20](#)
 Lists.h, [25](#)

CreateInput
 Lists.c, [20](#)
 Lists.h, [25](#)

CreateLInt
 Lists.c, [21](#)
 Lists.h, [25](#)

CreateNode
 Lists.c, [21](#)
 Lists.h, [25](#)

CreateSolution
 Lists.c, [21](#)
 Lists.h, [25](#)

CreateSystem
 Lists.c, [21](#)
 Lists.h, [25](#)

FindLGate
 Lists.c, [21](#)
 Lists.h, [25](#)

FindNodeID
 Lists.c, [21](#)
 Lists.h, [26](#)

FindNodeN
 Lists.c, [21](#)
 Lists.h, [26](#)

FindNodeVAL
 Lists.c, [22](#)
 Lists.h, [26](#)

Functions.c, [9](#)
 Attribution, [9](#)
 CompleteData, [10](#)
 CompleteStep, [10](#)
 Load1, [11](#)
 Load2, [11](#)
 MakeSolution, [12](#)
 Management, [12](#)
 Save, [12](#)
 ShowAll, [13](#)
 SolveStep, [13](#)

Functions.h, [14](#)
 Attribution, [14](#)
 CompleteData, [15](#)
 CompleteStep, [15](#)
 Load1, [15](#)
 Load2, [16](#)
 MakeSolution, [16](#)
 Management, [17](#)
 Save, [17](#)
 ShowAll, [18](#)
 SolveStep, [18](#)

Input, [5](#)
 Structures.h, [28](#)

LInt, [5](#)
 Structures.h, [28](#)

Lists.c, [18](#)
 AddInputEnd, [19](#)
 AddNodeEnd, [19](#)
 AddSolutionEnd, [19](#)
 AddSystemEnd, [20](#)
 CompareNode, [20](#)
 CopyLIntEnd, [20](#)

- CopyNodeEnd, 20
- CopyNodeList, 20
- CreateInput, 20
- CreateLInt, 21
- CreateNode, 21
- CreateSolution, 21
- CreateSystem, 21
- FindLGate, 21
- FindNodeID, 21
- FindNodeN, 21
- FindNodeVAL, 22
- PrintNode, 22
- ShowInput, 22
- ShowNode, 22
- ShowSolution, 22
- ShowSystem, 22
- Lists.h, 23
 - AddInputEnd, 23
 - AddNodeEnd, 24
 - AddSolutionEnd, 24
 - AddSystemEnd, 24
 - CompareNode, 24
 - CopyLIntEnd, 24
 - CopyNodeEnd, 24
 - CopyNodeList, 25
 - CreateInput, 25
 - CreateLInt, 25
 - CreateNode, 25
 - CreateSolution, 25
 - CreateSystem, 25
 - FindLGate, 25
 - FindNodeID, 26
 - FindNodeN, 26
 - FindNodeVAL, 26
 - PrintNode, 26
 - ShowInput, 26
 - ShowNode, 26
 - ShowSolution, 27
 - ShowSystem, 27
- Load1
 - Functions.c, 11
 - Functions.h, 15
- Load2
 - Functions.c, 11
 - Functions.h, 16
- MakeSolution
 - Functions.c, 12
 - Functions.h, 16
- Management
 - Functions.c, 12
 - Functions.h, 17
- Node, 6
 - Structures.h, 29
- PrintNode
 - Lists.c, 22
 - Lists.h, 26
- Save
 - Functions.c, 12
 - Functions.h, 17
- ShowAll
 - Functions.c, 13
 - Functions.h, 18
- ShowInput
 - Lists.c, 22
 - Lists.h, 26
- ShowNode
 - Lists.c, 22
 - Lists.h, 26
- ShowSolution
 - Lists.c, 22
 - Lists.h, 27
- ShowSystem
 - Lists.c, 22
 - Lists.h, 27
- Solution, 6
 - Structures.h, 29
- SolveStep
 - Functions.c, 13
 - Functions.h, 18
- Source.c, 27
- Structures.h, 27
 - bool, 28, 29
 - Input, 28
 - LInt, 28
 - Node, 29
 - Solution, 29
 - System, 29
 - Type, 29
- System, 7
 - Structures.h, 29
- Type
 - Structures.h, 29