

Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 3

Algorytm Dijkstry

autor	Michał Ślusarczyk
prowadzący	mgr. Grzegorz
rok akademicki	Kwiatkowski
kierunek	2019/2020
rodzaj studiów	Informatyka
semestr	SSI
sekcja	3
	2

termin oddania sprawozdania	2020-11-08
-----------------------------	------------

1 Treść zadania

Opis ogólny:

Napisać program generujący dla wskazanych przez użytkownika wierzchołków grafu tablice ze zbiorem decyzji określające przez które z sąsiednich wierzchołków grafu prowadzą najlepsze trasy do wybranych wybranych przez użytkownika wierzchołków oraz podając koszty (metryki) tych tras. W celu realizacji programu skorzystać z algorytmu Dijkstry.

Szczegółowe informacje na temat realizacji:

Opis grafu składa się z numerów wierzchołków, numerów sąsiadujących wierzchołków oraz metryk krawędzi grafu.

Opis podawany jest przez użytkownika za pomocą interfejsu który przeprowadza użytkownika przez proces konstrukcji grafu oraz określenia parametrów. Możliwe jest również wczytanie tych informacji z dwóch różnych plików. Możliwe są również warianty pośrednie (część informacji wczytanych z plików, a część z konsoli).

Postać informacji na temat grafu:

<nr. wierzchołka>: <nr. wierzchołka-sąsiada> <metryka>; ...

Postać informacji na temat tego jakie przejścia chcemy wyznaczyć:

<nr. wierzchołka>: <nr. wierzchołka-sąsiada>; ...

Postać informacji o rozwiązaniu powinna wyglądać w następujący sposób:

<dla wierzchołka>:

<wierzchołek docelowy>: → <przez wierzchołek-sąsiada> : <koszt trasy>

Metryki między wierzchołkami mogą się różnić w przeciwnych kierunkach. Założenie o braku metryki z wierzchołka A do A (wartość zostanie zignorowana),

Program powinien dopytywać się użytkownika o brakujące informacje w opisie grafu.

Wynik programu:

Opis grafu oraz wynik programu powinien zostać wyświetlony w interfejsie oraz zapisany w pliku o rozszerzeniu .txt wskazanym przez użytkownika (jeśli nie istnieje powinien zostać utworzony). Program powinien być uruchamiany z wykorzystaniem konsoli.

Interfejs:

Interfejsem programu wykorzystywanym do komunikacji z użytkownikiem jest konsola. W konsoli powinny wyświetlać się odpowiednie polecenia oraz pytania na które odpowiadać będzie użytkownik. Na tej podstawie program wyświetla opis grafu oraz wynik swojego działania.

Analiza zadania

Zagadnienie przedstawia zagadnienie wyznaczania optymalnej drogi z punktu A do punktu B w określonym układzie.

5.1 Struktury danych

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

5.2 Algorytmy

Program opiera się na algorytmie Dijkstry. Oprócz tego zastosowane zostało proste sortowanie przy wstawianiu co pozwoliło na automatyczne uszeregowanie danych.

6 Specyfikacja zewnętrzna

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

```
program.exe -G GRAPH.txt, -D DATA.txt, -R SOLUTION.txt
```

Pliki posiadają rozszerzenie tekstowe.

Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie zwrócenie błędu i przerwanie programu .

Podanie danych w nieodpowiedniej formule powoduje zwrócenie błędu i przerwanie programu .

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, wywoływany jest konstruktor tej klasy w którym uruchamiana jest metoda w której tworzone są obiekty potrzebne do obsługi określonych części programu.

Program podzielony został na etapy:
- User - Komunikacja z użytkownikiem i pozyskanie dokładnych informacji co ma wykonać program,

- Create – Wczytanie lub stworzenie danych wejściowych i grafu na których operuje program,

- Algorithm – realizacja algorytmu Dijkstry czyli stworzenie rozwiązania,

- Save – zapis danych wyjściowych do pliku wynikowego,

- Print – wyświetlenie danych na ekranie,

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. 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 algorytm Dijkstry 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. 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. Szczególnie kładąc nacisk na zarządzanie architekturą projektu w ten sposób aby był on przejrzysty i jego elementy ze sobą nie kolidowały szczególnie biorąc pod uwagę mechanizm dziedziczenia oraz obiektów składowych. Wiedza którą pozyskałem w zakresie synchronizacji elementów składowych programu przełoży się z pewnością na następne projekty.

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:Powszechnie błędny język](https://pl.wikipedia.org/wiki/Pomoc:Powszechnie_błędny_język)

Dodatek

Szczegółowy opis typów i funkcji

Algorytm Dijkstry

Wygenerowano przez Doxygen 1.8.20

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja klasy Algorithm	7
4.1.1 Opis szczegółowy	7
4.1.2 Dokumentacja funkcji składowych	7
4.1.2.1 CreateSolution()	7
4.1.2.2 Dijkstra()	8
4.1.2.3 PrepareSolution()	8
4.2 Dokumentacja klasy Borderer	9
4.2.1 Opis szczegółowy	9
4.2.2 Dokumentacja konstruktora i destruktora	9
4.2.2.1 Borderer()	9
4.2.3 Dokumentacja funkcji składowych	10
4.2.3.1 AddBordererByNumber()	10
4.2.3.2 AddBordererEnd()	10
4.2.3.3 DeleteBordererList()	10
4.2.3.4 FindBordererByNumber()	11
4.2.3.5 operator=()	11
4.3 Dokumentacja klasy Branch	11
4.3.1 Opis szczegółowy	12
4.3.2 Dokumentacja konstruktora i destruktora	12
4.3.2.1 Branch()	12
4.3.3 Dokumentacja funkcji składowych	12
4.3.3.1 AddBranchByEnd()	12
4.3.3.2 AddBranchEnd()	13
4.3.3.3 DeleteBranchList()	13
4.3.3.4 operator=()	13
4.4 Dokumentacja klasy Create	14
4.4.1 Opis szczegółowy	14
4.4.2 Dokumentacja funkcji składowych	14
4.4.2.1 CreateData()	14
4.4.2.2 CreateGraph()	15
4.4.2.3 LoadData()	15
4.4.2.4 LoadGraph()	15
4.5 Dokumentacja klasy Data	16

4.5.1 Opis szczegółowy	17
4.5.2 Dokumentacja konstruktora i destruktora	17
4.5.2.1 Data() [1/2]	17
4.5.2.2 Data() [2/2]	17
4.5.2.3 ~Data()	17
4.6 Dokumentacja klasy DataTops	18
4.6.1 Opis szczegółowy	18
4.6.2 Dokumentacja konstruktora i destruktora	18
4.6.2.1 DataTops()	18
4.6.3 Dokumentacja funkcji składowych	18
4.6.3.1 AddDataTopsByTop()	18
4.6.3.2 AddDataTopsEnd()	19
4.6.3.3 CreatePrimalList()	19
4.6.3.4 DeleteDataTopsList()	20
4.6.3.5 operator=()	20
4.6.4 Dokumentacja atrybutów składowych	20
4.6.4.1 pNext	20
4.7 Dokumentacja klasy Graph	20
4.7.1 Opis szczegółowy	21
4.7.2 Dokumentacja konstruktora i destruktora	21
4.7.2.1 Graph()	21
4.7.3 Dokumentacja funkcji składowych	21
4.7.3.1 AddGraphByTop()	21
4.7.3.2 AddGraphEnd()	22
4.7.3.3 DeleteGraphList()	22
4.7.3.4 operator=()	22
4.8 Dokumentacja klasy Management	23
4.8.1 Opis szczegółowy	23
4.8.2 Dokumentacja konstruktora i destruktora	24
4.8.2.1 Management() [1/2]	24
4.8.2.2 Management() [2/2]	24
4.8.3 Dokumentacja funkcji składowych	24
4.8.3.1 Coordinate()	24
4.9 Dokumentacja klasy Parameters	25
4.9.1 Dokumentacja konstruktora i destruktora	25
4.9.1.1 Parameters() [1/2]	25
4.9.1.2 Parameters() [2/2]	26
4.10 Dokumentacja klasy Predecessor	26
4.10.1 Opis szczegółowy	27
4.10.2 Dokumentacja konstruktora i destruktora	27
4.10.2.1 Predecessor()	27
4.10.3 Dokumentacja funkcji składowych	27

4.10.3.1 AddPredecessorByNumber()	27
4.10.3.2 AddPredecessorEnd()	27
4.10.3.3 CompletePredecessor()	28
4.10.3.4 DeletePredecessorList()	28
4.10.3.5 FindPredecessorByLowerCost()	28
4.10.3.6 FindPredecessorByNumber()	29
4.10.3.7 operator=()	29
4.10.3.8 PreparePredecessor()	30
4.11 Dokumentacja klasy Primal	30
4.11.1 Opis szczegółowy	31
4.11.2 Dokumentacja konstruktora i destruktora	31
4.11.2.1 Primal()	31
4.11.3 Dokumentacja funkcji składowych	31
4.11.3.1 AddPrimalByNumber()	31
4.11.3.2 AddPrimalEnd()	32
4.11.3.3 AddPrimalFront()	32
4.11.3.4 CheckPrimalByNumber()	33
4.11.3.5 CopyPrimal()	33
4.11.3.6 CopyPrimalList()	33
4.11.3.7 DeletePrimalByNumber()	34
4.11.3.8 DeletePrimalEnd()	34
4.11.3.9 DeletePrimalList()	34
4.11.3.10 FindPrimalByNumber()	35
4.11.3.11 operator=()	35
4.12 Dokumentacja klasy Print	35
4.12.1 Opis szczegółowy	36
4.13 Dokumentacja klasy PrintData	36
4.13.1 Opis szczegółowy	36
4.13.2 Dokumentacja konstruktora i destruktora	36
4.13.2.1 PrintData() [1/2]	37
4.13.2.2 PrintData() [2/2]	37
4.13.3 Dokumentacja funkcji składowych	37
4.13.3.1 PrintDijkstra()	37
4.14 Dokumentacja klasy PrintGraph	38
4.14.1 Opis szczegółowy	38
4.14.2 Dokumentacja konstruktora i destruktora	38
4.14.2.1 PrintGraph() [1/2]	38
4.14.2.2 PrintGraph() [2/2]	38
4.14.3 Dokumentacja funkcji składowych	39
4.14.3.1 PrintDijkstra()	39
4.15 Dokumentacja klasy PrintSolution	39
4.15.1 Opis szczegółowy	40

4.15.2 Dokumentacja konstruktora i destruktora	40
4.15.2.1 PrintSolution() [1/2]	40
4.15.2.2 PrintSolution() [2/2]	40
4.15.3 Dokumentacja funkcji składowych	40
4.15.3.1 PrintDijkstra()	40
4.16 Dokumentacja klasy Save	41
4.16.1 Opis szczegółowy	41
4.16.2 Dokumentacja funkcji składowych	41
4.16.2.1 ClearFile()	41
4.16.2.2 SaveData()	42
4.16.2.3 SaveGraph()	42
4.16.2.4 SaveSolution()	43
4.17 Dokumentacja klasy Solution	43
4.17.1 Opis szczegółowy	44
4.17.2 Dokumentacja konstruktora i destruktora	44
4.17.2.1 Solution()	44
4.17.3 Dokumentacja funkcji składowych	44
4.17.3.1 AddSolutionByStart()	44
4.17.3.2 AddSolutionEnd()	44
4.17.3.3 DeleteSolutionList()	45
4.17.3.4 operator=()	45
4.18 Dokumentacja klasy User	45
4.18.1 Opis szczegółowy	46
4.18.2 Dokumentacja funkcji składowych	46
4.18.2.1 Ask()	46
4.18.2.2 Calibration()	46
5 Dokumentacja plików	49
5.1 Dokumentacja pliku Algorithm.cpp	49
5.1.1 Opis szczegółowy	49
5.2 Dokumentacja pliku Algorithm.h	49
5.2.1 Opis szczegółowy	49
5.3 Dokumentacja pliku Borderer.cpp	49
5.3.1 Opis szczegółowy	50
5.4 Dokumentacja pliku Borderer.h	50
5.4.1 Opis szczegółowy	50
5.5 Dokumentacja pliku Branch.cpp	50
5.5.1 Opis szczegółowy	50
5.6 Dokumentacja pliku Branch.h	50
5.6.1 Opis szczegółowy	51
5.7 Dokumentacja pliku Create.cpp	51
5.7.1 Opis szczegółowy	51

5.8 Dokumentacja pliku Create.h	51
5.8.1 Opis szczegółowy	51
5.9 Dokumentacja pliku Data.cpp	51
5.9.1 Opis szczegółowy	51
5.10 Dokumentacja pliku Data.h	52
5.10.1 Opis szczegółowy	52
5.11 Dokumentacja pliku DataTops.cpp	52
5.11.1 Opis szczegółowy	52
5.12 Dokumentacja pliku DataTops.h	52
5.12.1 Opis szczegółowy	52
5.13 Dokumentacja pliku Graph.cpp	53
5.13.1 Opis szczegółowy	53
5.14 Dokumentacja pliku Graph.h	53
5.14.1 Opis szczegółowy	53
5.15 Dokumentacja pliku Management.cpp	53
5.15.1 Opis szczegółowy	53
5.16 Dokumentacja pliku Management.h	54
5.16.1 Opis szczegółowy	54
5.17 Dokumentacja pliku Operators.cpp	54
5.17.1 Opis szczegółowy	55
5.17.2 Dokumentacja funkcji	55
5.17.2.1 operator<<() [1/6]	55
5.17.2.2 operator<<() [2/6]	55
5.17.2.3 operator<<() [3/6]	56
5.17.2.4 operator<<() [4/6]	56
5.17.2.5 operator<<() [5/6]	56
5.17.2.6 operator<<() [6/6]	57
5.18 Dokumentacja pliku Operators.h	57
5.18.1 Opis szczegółowy	57
5.18.2 Dokumentacja funkcji	58
5.18.2.1 operator<<() [1/6]	58
5.18.2.2 operator<<() [2/6]	58
5.18.2.3 operator<<() [3/6]	58
5.18.2.4 operator<<() [4/6]	59
5.18.2.5 operator<<() [5/6]	59
5.18.2.6 operator<<() [6/6]	60
5.19 Dokumentacja pliku Parameters.cpp	60
5.19.1 Opis szczegółowy	60
5.20 Dokumentacja pliku Parameters.h	60
5.20.1 Opis szczegółowy	60
5.21 Dokumentacja pliku Predecessor.cpp	60
5.21.1 Opis szczegółowy	61

5.22 Dokumentacja pliku Predecessor.h	61
5.22.1 Opis szczegółowy	61
5.23 Dokumentacja pliku Primal.cpp	61
5.23.1 Opis szczegółowy	61
5.24 Dokumentacja pliku Primal.h	61
5.24.1 Opis szczegółowy	62
5.25 Dokumentacja pliku Print.h	62
5.25.1 Opis szczegółowy	62
5.26 Dokumentacja pliku PrintData.cpp	62
5.26.1 Opis szczegółowy	62
5.27 Dokumentacja pliku PrintData.h	62
5.27.1 Opis szczegółowy	62
5.28 Dokumentacja pliku PrintGraph.cpp	63
5.28.1 Opis szczegółowy	63
5.29 Dokumentacja pliku PrintGraph.h	63
5.29.1 Opis szczegółowy	63
5.30 Dokumentacja pliku PrintSolution.cpp	63
5.30.1 Opis szczegółowy	63
5.31 Dokumentacja pliku PrintSolution.h	63
5.31.1 Opis szczegółowy	64
5.32 Dokumentacja pliku Save.cpp	64
5.32.1 Opis szczegółowy	64
5.33 Dokumentacja pliku Save.h	64
5.33.1 Opis szczegółowy	64
5.34 Dokumentacja pliku Signs.h	64
5.34.1 Opis szczegółowy	64
5.35 Dokumentacja pliku Solution.cpp	65
5.35.1 Opis szczegółowy	65
5.36 Dokumentacja pliku Solution.h	65
5.36.1 Opis szczegółowy	65
5.37 Dokumentacja pliku Source.cpp	65
5.37.1 Opis szczegółowy	66
5.38 Dokumentacja pliku User.cpp	66
5.38.1 Opis szczegółowy	66
5.39 Dokumentacja pliku User.h	66
5.39.1 Opis szczegółowy	66
Indeks	67

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Algorithm	7
Branch	11
Create	14
Data	16
Print	35
PrintData	36
PrintGraph	38
PrintSolution	39
DataTops	18
Graph	20
Management	23
Parameters	25
Primal	30
Borderer	9
Predecessor	26
Save	41
Solution	43
User	45

Rozdział 2

Indeks klas

2.1 Lista klas

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

Algorithm	7
Borderer	9
Branch	11
Create	14
Data	16
DataTops	18
Graph	20
Management	23
Parameters	25
Predecessor	26
Primal	30
Print	35
PrintData	36
PrintGraph	38
PrintSolution	39
Save	41
Solution	43
User	45

Rozdział 3

Indeks plików

3.1 Lista plików

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

Algorithm.cpp	49
Algorithm.h	49
Borderer.cpp	49
Borderer.h	50
Branch.cpp	50
Branch.h	50
Create.cpp	51
Create.h	51
Data.cpp	51
Data.h	52
DataTops.cpp	52
DataTops.h	52
Graph.cpp	53
Graph.h	53
Management.cpp	53
Management.h	54
Operators.cpp	54
Operators.h	57
Parameters.cpp	60
Parameters.h	60
Predecessor.cpp	60
Predecessor.h	61
Primal.cpp	61
Primal.h	61
Print.h	62
PrintData.cpp	62
PrintData.h	62
PrintGraph.cpp	63
PrintGraph.h	63
PrintSolution.cpp	63
PrintSolution.h	63
Save.cpp	64
Save.h	64
Signs.h	64
Solution.cpp	65

Solution.h	65
Source.cpp	65
User.cpp	66
User.h	66

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy Algorithm

```
#include <Algorithm.h>
```

Metody publiczne

- bool `PrepareSolution` (`Data` *&)
- bool `Dijkstra` (`Data` *&)
- bool `CreateSolution` (`Solution` *&, `Predecessor` *)

4.1.1 Opis szczegółowy

Klasa odpowiedzialna za realizację algorytmu Dijkstry

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 `CreateSolution()`

```
bool Algorithm::CreateSolution (  
    Solution *& S,  
    Predecessor * P )
```

Publiczna metoda uzupełniająca rozwiązanie na podstawie działania algorytmu Dijkstry

Parametry

<i>DATA</i> Wskaźnik	na referencje do obiektu typu <code>Data</code> w którym przechowywane są wszystkie inne obiekty na których operuje program
<i>P</i>	Wskaźnik na listę obiektów typu <code>Predecessor</code> w której przechowywany jest wynik działania algorytmu Dijkstry

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.1.2.2 Dijkstra()

```
bool Algorithm::Dijkstra (
    Data *& DATA )
```

Publiczna metoda realizująca algorytm Dijkstry

Parametry

DATA	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.1.2.3 PrepareSolution()

```
bool Algorithm::PrepareSolution (
    Data *& DATA )
```

Publiczna metoda przygotowująca strukturę listy która później ma przechowywać rozwiązanie

Parametry

DATA	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

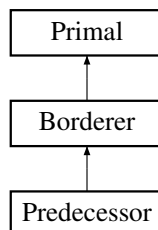
Dokumentacja dla tej klasy została wygenerowana z plików:

- [Algorithm.h](#)
- [Algorithm.cpp](#)

4.2 Dokumentacja klasy Borderer

```
#include <Borderer.h>
```

Diagram dziedziczenia dla Borderer



Metody publiczne

- [Borderer](#) ()
- [Borderer](#) & [operator=](#) (const [Borderer](#) &)
- bool [DeleteBordererList](#) ()

Statyczne metody publiczne

- static bool [AddBordererEnd](#) ([Borderer](#) *&, [Borderer](#) *)
- static bool [AddBordererByNumber](#) ([Borderer](#) *&, [Borderer](#) *)
- static [Borderer](#) * [FindBordererByNumber](#) ([Borderer](#) *, long int)

Atrybuty publiczne

- double [metrics](#)
Odległość do danego sąsiedniego wierzchołka.
- [Borderer](#) * [pNext](#)
Wskaźnik na następny element.
- [Borderer](#) * [pPrev](#)
Wskaźnik na poprzedni element.

4.2.1 Opis szczegółowy

Klasa będąca odpowiedzialna za listę z sąsiadami wierzchołka startowego

4.2.2 Dokumentacja konstruktora i destruktor

4.2.2.1 Borderer()

```
Borderer::Borderer ( )
```

Konstruktor domyślny klasy [Borderer](#)

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 AddBordererByNumber()

```
bool Borderer::AddBordererByNumber (
    Borderer *& LB,
    Borderer * B ) [static]
```

Publiczna metoda dodająca element typu [Borderer](#) z uwzględnieniem numeru sąsiedniego wierzchołka

Parametry

LS	Wskaźnik na referencje typu Borderer
S	Wskaźnik na obiekt typu Borderer

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.2.3.2 AddBordererEnd()

```
bool Borderer::AddBordererEnd (
    Borderer *& LB,
    Borderer * B ) [static]
```

Publiczna metoda dodająca element typu [Borderer](#) na koniec listy

Parametry

LB	Wskaźnik na referencje typu Solution
B	Wskaźnik na obiekt typu Solution

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.2.3.3 DeleteBordererList()

```
bool Borderer::DeleteBordererList ( )
```

Publiczna metoda do usuwania listy typu [Borderer](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.2.3.4 FindBordererByNumber()

```
Borderer * Borderer::FindBordererByNumber (
    Borderer * B,
    long int N ) [static]
```

Publiczna metoda dodająca szukającą element w liście typu [Borderer](#) o zadanym numerze

Parametry

<i>B</i>	Wskaźnik na listę typu Borderer
<i>N</i>	Numer szukanego wierzchołka

Zwraca

Zwraca wskaźnik na szukany element lub nullptr

4.2.3.5 operator=()

```
Borderer & Borderer::operator= (
    const Borderer & B )
```

Przeciążony operator przypisania

Parametry

<i>S</i>	Referencja do stałego obiektu typu Borderer
----------	---

Zwraca

Referencja do elementu typu [Borderer](#) pozwalająca na kaskadowe wywołanie

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Borderer.h](#)
- [Borderer.cpp](#)

4.3 Dokumentacja klasy Branch

```
#include <Branch.h>
```

Metody publiczne

- [Branch](#) ()
- [Branch](#) & [operator=](#) (const [Branch](#) &)
- bool [DeleteBranchList](#) ()

Statyczne metody publiczne

- static bool [AddBranchEnd](#) ([Branch](#) *&, [Branch](#) *)
- static bool [AddBranchByEnd](#) ([Branch](#) *&, [Branch](#) *)

Atrybuty publiczne

- [Borderer](#) * [end](#)
Wskaźnik na element typu [Borderer](#) będący wierzchołkiem końcowym.
- [Primal](#) * [way](#)
Wskaźnik na listę typu [Primal](#) przechowującą trasę do węzła wynikowego.
- [Branch](#) * [pNext](#)
Wskaźnik na następny element.
- [Branch](#) * [pPrev](#)
Wskaźnik na poprzedni element.

4.3.1 Opis szczegółowy

Klasa będąca odpowiedzialna za listę typu [Branch](#)

4.3.2 Dokumentacja konstruktora i destruktor

4.3.2.1 [Branch](#)()

```
Branch::Branch ( )
```

Konstruktor domyślny klasy [Branch](#)

4.3.3 Dokumentacja funkcji składowych

4.3.3.1 [AddBranchByEnd](#)()

```
bool Branch::AddBranchByEnd (  
    Branch *& LB,  
    Branch * B ) [static]
```

Publiczna metoda dodająca element typu [Branch](#) z uwzględnieniem numeru wiechołka końcowego

Parametry

<i>LB</i>	Wskaźnik na referencje typu Branch
<i>B</i>	Wskaźnik na obiekt typu Branch

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.3.3.2 AddBranchEnd()

```
bool Branch::AddBranchEnd (
    Branch *& LB,
    Branch * B ) [static]
```

Publiczna metoda dodająca element typu [Branch](#) na koniec listy

Parametry

<i>LB</i>	Wskaźnik na referencje typu Branch
<i>B</i>	Wskaźnik na obiekt typu Branch

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.3.3.3 DeleteBranchList()

```
bool Branch::DeleteBranchList ( )
```

Publiczna metoda do usuwania listy typu [Branch](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.3.3.4 operator=()

```
Branch & Branch::operator= (
    const Branch & B )
```

Przeciążony operator przypisania

Parametry

<i>B</i>	Referencja do stałego obiektu typu Solution
----------	---

Zwraca

Referencja do elementu typu [Solution](#) pozwalająca na kaskadowe wywołanie

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Branch.h](#)
- [Branch.cpp](#)

4.4 Dokumentacja klasy Create

```
#include <Create.h>
```

Metody publiczne

- bool [LoadGraph](#) ([Data](#) *&)
- bool [CreateGraph](#) ([Data](#) *&)
- bool [LoadData](#) ([Data](#) *&)
- bool [CreateData](#) ([Data](#) *&)

4.4.1 Opis szczegółowy

Klasa odpowiedzialna za wczytanie lub stworzenie w oparciu o komunikację z użytkownikiem danych na których ma operować program

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 CreateData()

```
bool Create::CreateData (  
    Data *& DATA )
```

Metoda która tworzy dane na podstawie informacji od użytkownika

Parametry

<i>DATA</i>	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.4.2.2 CreateGraph()

```
bool Create::CreateGraph (
    Data *& DATA )
```

Metoda która tworzy graf na podstawie informacji od użytkownika

Parametry

DATA	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	--

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.4.2.3 LoadData()

```
bool Create::LoadData (
    Data *& DATA )
```

Metoda która wczytuje dane z pliku wskazanego przez użytkownika

Parametry

DATA	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	--

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.4.2.4 LoadGraph()

```
bool Create::LoadGraph (
    Data *& DATA )
```

Metoda która wczytuje graf z pliku wskazanego przez użytkownika

Parametry

<i>DATA</i>	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

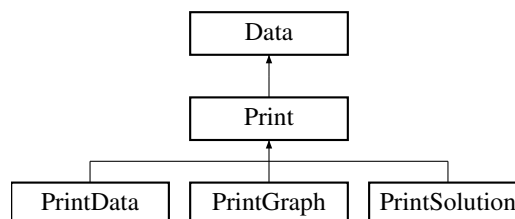
Dokumentacja dla tej klasy została wygenerowana z plików:

- [Create.h](#)
- [Create.cpp](#)

4.5 Dokumentacja klasy Data

```
#include <Data.h>
```

Diagram dziedziczenia dla Data



Metody publiczne

- [Data](#) ()
- [Data](#) ([Parameters](#) *[P](#), [Graph](#) *[G](#), [DataTops](#) *[D](#), [Solution](#) *[S](#))
- [~Data](#) ()

Atrybuty publiczne

- [Parameters](#) * [P](#)
Atrybut klasy [Parameters](#) przechowujący dane na temat funkcjonalności które mają być użyte.
- [Graph](#) * [G](#)
Atrybut klasy [Graph](#) w którym przechowywany jest opis grafu.
- [DataTops](#) * [D](#)
Atrybut klasy [DataTops](#) w którym przechowywane są dane wejściowe.
- [Solution](#) * [S](#)
Atrybut klasy [Solution](#) w którym przechowywane jest rozwiązanie.

4.5.1 Opis szczegółowy

Klasa do przechowywania wszystkich struktur potrzebnych do realizacji algorytmu Dijkstry

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 Data() [1/2]

```
Data::Data ( )
```

Konstruktor domyślny klasy [Data](#)

4.5.2.2 Data() [2/2]

```
Data::Data (
    Parameters *p,
    Graph *g,
    DataTops *d,
    Solution *s )
```

Wieloargumentowy konstruktor klasy [Data](#) do zapisu danych z klasy [Management](#)

Parametry

<i>p</i>	Wskaźnik na referencje do obiektu typu Parameters w którym przechowywane są dane na temat działania programu
<i>g</i>	Wskaźnik na referencje do obiektu typu Graph w którym przechowywany jest graf
<i>d</i>	Wskaźnik na referencje do obiektu typu DataTops w którym przechowywane są dane wejściowe
<i>s</i>	Wskaźnik na referencje do obiektu typu Solution w którym przechowywane jest rozwiązanie

4.5.2.3 ~Data()

```
Data::~Data ( )
```

Destruktor za którego pomocą jest zwalniana cała dynamicznie zarezerwowana pamięć w trakcie działania programu

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Data.h](#)
- [Data.cpp](#)

4.6 Dokumentacja klasy DataTops

```
#include <DataTops.h>
```

Metody publiczne

- [DataTops](#) & [operator=](#) (const [DataTops](#) &)
- [DataTops](#) ()
- bool [DeleteDataTopsList](#) ()

Statyczne metody publiczne

- static bool [AddDataTopsEnd](#) ([DataTops](#) *&, [DataTops](#) *)
- static bool [AddDataTopsByTop](#) ([DataTops](#) *&, [DataTops](#) *)
- static bool [CreatePrimalList](#) ([Primal](#) *&, [DataTops](#) *)

Atrybuty publiczne

- [Primal](#) * [top](#)
- [Primal](#) * [Dtops](#)
<Wskaźnik na wierzchołek startowy
- [DataTops](#) * [pNext](#)
<Wskaźnik na listę wierzchołków do których chcemy wyznaczyć trasę
- [DataTops](#) * [pPrev](#)
Wskaźnik na poprzedni element.

Przyjaciele

- class [Primal](#)

4.6.1 Opis szczegółowy

Klasa będąca odpowiedzialna za listę z danymi wejściowymi

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 DataTops()

```
DataTops::DataTops ( )
```

Konstruktor domyślny klasy [DataTops](#)

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 AddDataTopsByTop()

```
bool DataTops::AddDataTopsByTop (
    DataTops *& LD,
    DataTops * D ) [static]
```

Publiczna metoda dodająca element typu [DataTops](#) z uwzględnieniem numeru wierzchołka startowego

Parametry

<i>LD</i>	Wskaźnik na referencje typu DataTops
<i>D</i>	Wskaźnik na obiekt typu DataTops

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.6.3.2 AddDataTopsEnd()

```
bool DataTops::AddDataTopsEnd (
    DataTops * & LD,
    DataTops * D ) [static]
```

Publiczna metoda dodająca element typu [DataTops](#) na koniec listy

Parametry

<i>LD</i>	Wskaźnik na referencje typu DataTops
<i>D</i>	Wskaźnik na obiekt typu DataTops

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.6.3.3 CreatePrimalList()

```
bool DataTops::CreatePrimalList (
    Primal * & P,
    DataTops * D ) [static]
```

Publiczna metoda tworząca listę poprzedników

Parametry

<i>P</i>	Wskaźnik na referencje typu Primal
<i>D</i>	Wskaźnik na listę typu DataTops

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.6.3.4 DeleteDataTopsList()

```
bool DataTops::DeleteDataTopsList ( )
```

Publiczna metoda do usuwania listy typu [DataTops](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.6.3.5 operator=()

```
DataTops & DataTops::operator= (
    const DataTops & D )
```

Przeciążony operator przypisania

Parametry

<i>D</i>	Referencja do stałego obiektu typu DataTops
----------	---

Zwraca

Referencja do elementu typu [DataTops](#) pozwalająca na kaskadowe wywołanie

4.6.4 Dokumentacja atrybutów składowych

4.6.4.1 pNext

```
DataTops* DataTops::pNext
```

<Wskaźnik na listę wierzchołków do których chcemy wyznaczyć trasę

Wskaźnik na następny element

Dokumentacja dla tej klasy została wygenerowana z plików:

- [DataTops.h](#)
- [DataTops.cpp](#)

4.7 Dokumentacja klasy Graph

```
#include <Graph.h>
```

Metody publiczne

- `Graph ()`
- `Graph & operator= (const Graph &)`
- `bool DeleteGraphList ()`

Statyczne metody publiczne

- `static bool AddGraphEnd (Graph *&, Graph *)`
- `static bool AddGraphByTop (Graph *&, Graph *)`

Atrybuty publiczne

- `Primal * top`
- `Borderer * BTops`
- `Graph * pNext`
- `Graph * pPrev`

Przyjaciele

- `class Borderer`

4.7.1 Opis szczegółowy

Klasa będąca odpowiedzialna za graf

4.7.2 Dokumentacja konstruktora i destruktor

4.7.2.1 Graph()

```
Graph::Graph ( )
```

Konstruktor domyślny klasy Graf

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 AddGraphByTop()

```
bool Graph::AddGraphByTop (
    Graph *& LG,
    Graph * G ) [static]
```

Publiczna metoda dodająca element typu `Graph` z uwzględnieniem numeru wierzchołka startowego

Parametry

<i>LG</i>	Wskaźnik na referencje do listy typu Graph
<i>G</i>	Wskaźnik na obiekt typu Graph

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.7.3.2 AddGraphEnd()

```
bool Graph::AddGraphEnd (
    Graph *& LG,
    Graph * G ) [static]
```

Publiczna metoda dodająca element typu [Graph](#) na koniec listy

Parametry

<i>LG</i>	Wskaźnik na referencje do listy typu Graph
<i>G</i>	Wskaźnik na obiekt typu Graph

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.7.3.3 DeleteGraphList()

```
bool Graph::DeleteGraphList ( )
```

Publiczna metoda do usuwania listy typu [Graph](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.7.3.4 operator=()

```
Graph & Graph::operator= (
    const Graph & G )
```

Przeciążony operator przypisania

Parametry

G	Referencja do stałego obiektu typu Graph
---	--

Zwraca

Referencja do elementu typu [Graph](#) pozwalająca na kaskadowe wywołanie

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Graph.h](#)
- [Graph.cpp](#)

4.8 Dokumentacja klasy Management

```
#include <Management.h>
```

Metody publiczne

- [Management](#) ()
- [Management](#) (std::string p[])
- bool [Coordinate](#) ([Data](#) *&)

Atrybuty chronione

- [Parameters](#) * P
Atrybut klasy [Parameters](#) przechowujący dane na temat funkcjonalności które mają być użyte.
- [Graph](#) * G
Atrybut klasy [Graph](#) w którym przechowywany jest opis grafu.
- [DataTops](#) * D
Atrybut klasy [DataTops](#) w którym przechowywane są dane wejściowe.
- [Solution](#) * S
Atrybut klasy [Solution](#) w którym przechowywane jest rozwiązanie.

Przyjaciele

- class [Parameters](#)
Zaprzyżyczenie z klasą zawierającą parametry układu.
- class [Print](#)
Zaprzyżyczenie z klasą abstrakcyjną [Print](#) wykorzystywaną podczas wyświetlania.

4.8.1 Opis szczegółowy

Klasa odpowiedzialna za obsługę procesu wykonywania zadania programistycznego

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 Management() [1/2]

```
Management::Management ( )
```

Konstruktor domyślny klasy [Management](#)

4.8.2.2 Management() [2/2]

```
Management::Management (
    std::string p[] )
```

Konstruktor z argumentem będącym ścieżką do plików wejścia/wyjścia

Parametry

<i>p</i>	Tablica typu string przechowująca ścieżki do plików wejścia/wyjścia
----------	---

4.8.3 Dokumentacja funkcji składowych

4.8.3.1 Coordinate()

```
bool Management::Coordinate (
    Data *& DATA )
```

Konstruktor z argumentem będącym ścieżką do plików wejścia/wyjścia

Parametry

<i>DATA</i>	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Management.h](#)
- [Management.cpp](#)

4.9 Dokumentacja klasy Parameters

Metody publiczne

- [Parameters](#) ()
- [Parameters](#) (std::string[])

Atrybuty publiczne

- std::string [paths](#) [3]
Atrybut będący tablicą typu string zawierający ścieżki do pliku.
- bool [Qgraph](#)
Atrybut typu bool zawierający informacje czy ścieżka do pliku z grafem została wczytana z konsoli.
- bool [Qgrapho](#)
Atrybut typu bool zawierający informacje czy użytkownik chce opisać graf sam.
- bool [QIO](#)
Atrybut typu bool zawierający informacje czy ścieżka do pliku z danymi została wczytana z konsoli.
- bool [QIOo](#)
Atrybut typu bool zawierający informacje czy użytkownik chce podać dane ręcznie.
- bool [Qsave](#)
Atrybut typu bool zawierający informacje czy ścieżka do pliku gdzie ma być zapisane rozwiązanie została wczytana z konsoli.
- bool [Qsaveg](#)
Atrybut typu bool zawierający informacje czy zapisać graf.
- bool [Qsaved](#)
Atrybut typu bool zawierający informacje czy zapisać dane.
- bool [Qsaves](#)
Atrybut typu bool zawierający informacje czy zapisać rozwiązanie.
- bool [Qshowg](#)
Atrybut typu bool zawierający informacje czy wyświetlać graf.
- bool [Qshowd](#)
Atrybut typu bool zawierający informacje czy wyświetlać dane.
- bool [Qshows](#)
Atrybut typu bool zawierający informacje czy wyświetlać rozwiązanie.

Przyjaciele

- class [Management](#)
Zaprzyjaźnienie z klasą służącą do zarządzania programem.

4.9.1 Dokumentacja konstruktora i destruktora

4.9.1.1 Parameters() [1/2]

```
Parameters::Parameters ( )
```

Konstruktor domyślny klasy [Parameters](#)

4.9.1.2 Parameters() [2/2]

```
Parameters::Parameters (
    std::string p[] )
```

Jednoargumentowy konstruktor klasy [Parameters](#) który przypisuje sobie przekazane ścieżki do plików

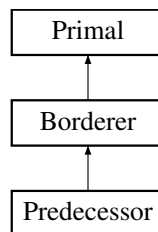
Dokumentacja dla tej klasy została wygenerowana z plików:

- [Parameters.h](#)
- [Parameters.cpp](#)

4.10 Dokumentacja klasy Predecessor

```
#include <Predecessor.h>
```

Diagram dziedziczenia dla Predecessor



Metody publiczne

- [Predecessor](#) ()
- [Predecessor](#) & operator= (const [Predecessor](#) &)
- bool [DeletePredecessorList](#) ()

Statyczne metody publiczne

- static bool [AddPredecessorEnd](#) ([Predecessor](#) *&, [Predecessor](#) *)
- static bool [AddPredecessorByNumber](#) ([Predecessor](#) *&, [Predecessor](#) *)
- static bool [PreparePredecessor](#) ([Predecessor](#) *&, [Primal](#) *, long int)
- static [Predecessor](#) * [FindPredecessorByLowerCost](#) ([Predecessor](#) *, [Primal](#) *)
- static [Predecessor](#) * [FindPredecessorByNumber](#) ([Predecessor](#) *, long int N)
- static bool [CompletePredecessor](#) ([Predecessor](#) *&, [Primal](#) *, long int, [Graph](#) *)

Atrybuty publiczne

- long int [predecessor](#)
Numer poprzedniego wierzchołka na trasie.
- [Predecessor](#) * [pNext](#)
Wskaźnik na następny element.
- [Predecessor](#) * [pPrev](#)
Wskaźnik na poprzedni element.

4.10.1 Opis szczegółowy

Klasa będąca odpowiedzialną za przechowywanie efektu działania algorytmu Dijkstry

4.10.2 Dokumentacja konstruktora i destruktora

4.10.2.1 Predecessor()

```
Predecessor::Predecessor ( )
```

Konstruktor domyślny klasy [Solution](#)

4.10.3 Dokumentacja funkcji składowych

4.10.3.1 AddPredecessorByNumber()

```
bool Predecessor::AddPredecessorByNumber (
    Predecessor *& LP,
    Predecessor * P ) [static]
```

Publiczna metoda dodająca element typu [Predecessor](#) w zależności od wartości od numeru

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Predecessor
<i>P</i>	Wskaźnik na obiekt typu Predecessor

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.10.3.2 AddPredecessorEnd()

```
bool Predecessor::AddPredecessorEnd (
    Predecessor *& LP,
    Predecessor * P ) [static]
```

Publiczna metoda dodająca element typu [Predecessor](#) na koniec listy

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Predecessor
<i>P</i>	Wskaźnik na obiekt typu Predecessor

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.10.3.3 CompletePredecessor()

```
bool Predecessor::CompletePredecessor (
    Predecessor * & LP,
    Primal * P,
    long int N,
    Graph * G ) [static]
```

Publiczna metoda sprawdzająca czy dana trasa jest lepsza i ewentualnie aktualizująca dane

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Predecessor
<i>P</i>	Wskaźnik na rozważany element
<i>N</i>	Numer wierzchołka startowego
<i>G</i>	Wskaźnik na graf

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.10.3.4 DeletePredecessorList()

```
bool Predecessor::DeletePredecessorList ( )
```

Publiczna metoda usuwająca listę typu [Predecessor](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.10.3.5 FindPredecessorByLowerCost()

```
Predecessor * Predecessor::FindPredecessorByLowerCost (
    Predecessor * LP,
    Primal * P ) [static]
```

Publiczna metoda znajdująca najmniejszy element listy pod względem odległości

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Predecessor
<i>P</i>	Wskaźnik na obiekt typu Primal

Zwraca

Zwraca wskaźnik na element o najmniejszym koszcie

4.10.3.6 FindPredecessorByNumber()

```
Predecessor * Predecessor::FindPredecessorByNumber (
    Predecessor * LP,
    long int N ) [static]
```

Publiczna metoda znajdujaca element listy po numerze

Parametry

<i>LP</i>	Wskaźnik na listę typu Predecessor
<i>N</i>	Numer wierzchołka

Zwraca

Zwraca wskaźnik na znaleziony element lub nullptr

4.10.3.7 operator=()

```
Predecessor & Predecessor::operator= (
    const Predecessor & P )
```

Przeciążony operator przypisania

Parametry

<i>S</i>	Referencja do stałego obiektu typu Predecessor
----------	--

Zwraca

Referencja do elementu typu [Predecessor](#) pozwalająca na kaskadowe wywołanie

4.10.3.8 PreparePredecessor()

```
bool Predecessor::PreparePredecessor (
    Predecessor * & P,
    Primal * PR,
    long int N ) [static]
```

Publiczna metoda przygotowująca liste typu Predecessor do uzupełnienia

Parametry

<i>P</i>	Wskaźnik na listę typu Predecessor
<i>PR</i>	Wskaźnik na listę typu Primal przechowującą numery wszystkich wierzchołków w grafie
<i>N</i>	Numer wierzchołka startowego

Zwraca

Referencja do elementu typu [Solution](#) pozwalająca na kaskadowe wywołanie

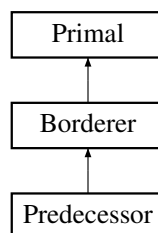
Dokumentacja dla tej klasy została wygenerowana z plików:

- [Predecessor.h](#)
- [Predecessor.cpp](#)

4.11 Dokumentacja klasy Primal

```
#include <Primal.h>
```

Diagram dziedziczenia dla Primal



Metody publiczne

- [Primal \(\)](#)
- [Primal & operator= \(const Primal &\)](#)
- [bool DeletePrimalList \(\)](#)
- [bool DeletePrimalEnd \(Primal * &\)](#)

Statyczne metody publiczne

- static bool `AddPrimalEnd` (`Primal` *&, `Primal` *)
- static bool `AddPrimalFront` (`Primal` *&, `Primal` *)
- static bool `AddPrimalByNumber` (`Primal` *&, `Primal` *)
- static bool `CopyPrimalList` (`Primal` *&, `Primal` *)
- static bool `CopyPrimal` (`Primal` *&, `Primal` *)
- static bool `DeletePrimalByNumber` (`Primal` *&, long int)
- static `Primal` * `FindPrimalByNumber` (`Primal` *, long int)
- static bool `CheckPrimalByNumber` (`Primal` *, long int)

Atrybuty publiczne

- long int `number`
Numer zadania.
- `Primal` * `pNext`
Wskaźnik na następny element.
- `Primal` * `pPrev`
Wskaźnik na poprzedni element.

4.11.1 Opis szczegółowy

Klasa będąca odpowiedzialna za liste typu `Primal`

4.11.2 Dokumentacja konstruktora i destruktor

4.11.2.1 `Primal()`

```
Primal::Primal ( )
```

Konstruktor domyślny klasy `Primal`

4.11.3 Dokumentacja funkcji składowych

4.11.3.1 `AddPrimalByNumber()`

```
bool Primal::AddPrimalByNumber (
    Primal *& LP,
    Primal * P ) [static]
```

Publiczna metoda dodająca element typu `Primal` w zależności od numeru

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Primal
<i>P</i>	Wskaźnik na obiekt typu Primal

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.2 AddPrimalEnd()

```
bool Primal::AddPrimalEnd (
    Primal * & LP,
    Primal * P ) [static]
```

Publiczna metoda dodająca element typu [Primal](#) na koniec listy

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Primal
<i>P</i>	Wskaźnik na obiekt typu Primal

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.3 AddPrimalFront()

```
bool Primal::AddPrimalFront (
    Primal * & LP,
    Primal * P ) [static]
```

Publiczna metoda dodająca element typu [Primal](#) na początek listy

Parametry

<i>LP</i>	Wskaźnik na referencje do listy typu Primal
<i>P</i>	Wskaźnik na obiekt typu Primal

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.4 CheckPrimalByNumber()

```
bool Primal::CheckPrimalByNumber (
    Primal * P,
    long int N ) [static]
```

Publiczna metoda do sprawdzenia czy element o takim numerze jest w liście typu [Primal](#)

Parametry

<i>P</i>	Wskaźnik na listę typu Primal
<i>N</i>	Numer wierzchołka do sprawdzenia

Zwraca

Zwraca informację czy element został znaleziony

4.11.3.5 CopyPrimal()

```
bool Primal::CopyPrimal (
    Primal *& P,
    Primal * CP ) [static]
```

Publiczna metoda do skopiowania elementu typu [Primal](#)

Parametry

<i>P</i>	Wskaźnik na referencje do miejsca do którego chcemy kopiować
<i>CP</i>	Wskaźnik na element typu Primal do skopiowania

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.6 CopyPrimalList()

```
bool Primal::CopyPrimalList (
    Primal *& P,
    Primal * CP ) [static]
```

Publiczna metoda do skopiowania elementu typu [Primal](#)

Parametry

<i>P</i>	Wskaźnik na referencje do miejsca do którego chcemy kopiować
<i>CP</i>	Wskaźnik na listę typu Primal do skopiowania

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.7 DeletePrimalByNumber()

```
bool Primal::DeletePrimalByNumber (
    Primal * & P,
    long int N ) [static]
```

Publiczna metoda usuwająca element typu [Primal](#) w zależności od numeru

Parametry

<i>P</i>	Wskaźnik na referencje do listy typu Primal
<i>N</i>	Numer wierzchołka do usunięcia

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.8 DeletePrimalEnd()

```
bool Primal::DeletePrimalEnd (
    Primal * & P )
```

Publiczna metoda do usuwania ostatniego elementu listy typu [Primal](#)

Parametry

<i>P</i>	Wskaźnik na referencje do listy typu Primal
----------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.9 DeletePrimalList()

```
bool Primal::DeletePrimalList ( )
```

Publiczna metoda do usuwania listy typu [Primal](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.11.3.10 FindPrimalByNumber()

```
Primal * Primal::FindPrimalByNumber (
    Primal * P,
    long int N ) [static]
```

Publiczna metoda do szukania elementu o określonym numerze w liście typu [Primal](#)

Parametry

<i>P</i>	Wskaźnik na liście typu Primal
<i>N</i>	Numer wierzchołka do znalezienia

Zwraca

Zwraca wskaźnik na szukany element

4.11.3.11 operator=()

```
Primal & Primal::operator= (
    const Primal & P )
```

Przeciążony operator przypisania

Parametry

<i>P</i>	Referencja do stałego obiektu typu Primal
----------	---

Zwraca

Referencja do elementu typu [Primal](#) pozwalająca na kaskadowe wywołanie

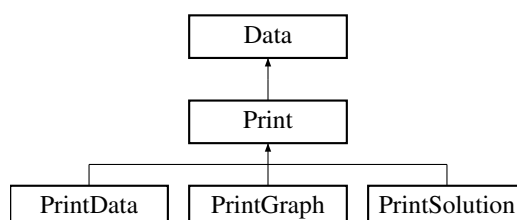
Dokumentacja dla tej klasy została wygenerowana z plików:

- [Primal.h](#)
- [Primal.cpp](#)

4.12 Dokumentacja klasy Print

```
#include <Print.h>
```

Diagram dziedziczenia dla Print



Metody publiczne

- virtual bool **PrintDijkstra** ()=0

Dodatkowe Dziedziczone Składowe

4.12.1 Opis szczegółowy

Klasa abstrakcyjna pozwalająca po której dziedziczą kalsy zawierające metody do wyświetlania konkretnych danych na ekranie

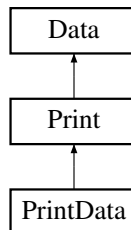
Dokumentacja dla tej klasy została wygenerowana z pliku:

- [Print.h](#)

4.13 Dokumentacja klasy PrintData

```
#include <PrintData.h>
```

Diagram dziedziczenia dla PrintData



Metody publiczne

- [PrintData](#) ()
- [PrintData](#) (Data *)
- bool [PrintDijkstra](#) ()

Dodatkowe Dziedziczone Składowe

4.13.1 Opis szczegółowy

Klasa odpowiedzialna za wyświetlanie danych na ekranie

4.13.2 Dokumentacja konstruktora i destruktor

4.13.2.1 PrintData() [1/2]

```
PrintData::PrintData ( )
```

Konstruktor domyślny klasy [PrintData](#)

4.13.2.2 PrintData() [2/2]

```
PrintData::PrintData (
    Data * )
```

Jednoargumentowy konstruktor klasy [PrintData](#)

Parametry

DATA	Wskaźnik na obiekt typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
------	--

4.13.3 Dokumentacja funkcji składowych

4.13.3.1 PrintDijkstra()

```
bool PrintData::PrintDijkstra ( ) [virtual]
```

Publiczna metoda wyświetlająca dane wejściowe na ekranie

Parametry

DATA	Wskaźnik na obiekt typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
------	--

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

Implementuje [Print](#).

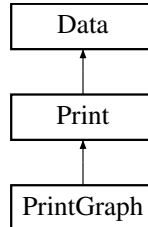
Dokumentacja dla tej klasy została wygenerowana z plików:

- [PrintData.h](#)
- [PrintData.cpp](#)

4.14 Dokumentacja klasy PrintGraph

```
#include <PrintGraph.h>
```

Diagram dziedziczenia dla PrintGraph



Metody publiczne

- [PrintGraph](#) ()
- [PrintGraph](#) ([Data](#) *)
- bool [PrintDijkstra](#) ()

Dodatkowe Dziedziczone Składowe

4.14.1 Opis szczegółowy

Klasa odpowiedzialna za wyświetlanie grafu na ekranie

4.14.2 Dokumentacja konstruktora i destruktora

4.14.2.1 [PrintGraph](#)() [1/2]

```
PrintGraph::PrintGraph ( )
```

Konstruktor domyślny klasy [PrintGraph](#)

4.14.2.2 [PrintGraph](#)() [2/2]

```
PrintGraph::PrintGraph (  
    Data * DATA )
```

Jednoargumentowy konstruktor klasy [PrintGraph](#)

Parametry

DATA	Wskaźnik na obiekt typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
------	--

4.14.3 Dokumentacja funkcji składowych

4.14.3.1 PrintDijkstra()

```
bool PrintGraph::PrintDijkstra ( ) [virtual]
```

Publiczna metoda wyświetlająca graf na ekranie

Parametry

DATA	Wskaźnik na obiekt typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
------	--

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

Implementuje [Print](#).

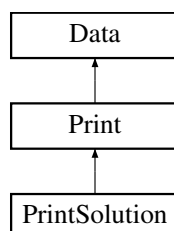
Dokumentacja dla tej klasy została wygenerowana z plików:

- [PrintGraph.h](#)
- [PrintGraph.cpp](#)

4.15 Dokumentacja klasy PrintSolution

```
#include <PrintSolution.h>
```

Diagram dziedziczenia dla PrintSolution



Metody publiczne

- [PrintSolution](#) ()
- [PrintSolution](#) ([Data](#) *)
- bool [PrintDijkstra](#) ()

Dodatkowe Dziedziczone Składowe

4.15.1 Opis szczegółowy

Klasa odpowiedzialna za wyświetlanie rozwiązania na ekranie

4.15.2 Dokumentacja konstruktora i destruktora

4.15.2.1 [PrintSolution](#)() [1/2]

```
PrintSolution::PrintSolution ( )
```

Konstruktor domyślny klasy [PrintSolution](#)

4.15.2.2 [PrintSolution](#)() [2/2]

```
PrintSolution::PrintSolution (
    Data * DATA )
```

Jednoargumentowy konstruktor klasy [PrintSolution](#)

Parametry

<i>DATA</i>	Wskaźnik na obiekt typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	--

4.15.3 Dokumentacja funkcji składowych

4.15.3.1 [PrintDijkstra](#)()

```
bool PrintSolution::PrintDijkstra ( ) [virtual]
```

Publiczna metoda wyświetlająca rozwiązanie na ekranie

Parametry

DATA	Wskaźnik na obiekt typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
------	--

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

Implementuje [Print](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

- [PrintSolution.h](#)
- [PrintSolution.cpp](#)

4.16 Dokumentacja klasy Save

```
#include <Save.h>
```

Metody publiczne

- bool [ClearFile](#) ([Data](#) *)
- bool [SaveGraph](#) ([Data](#) *)
- bool [SaveData](#) ([Data](#) *)
- bool [SaveSolution](#) ([Data](#) *)

4.16.1 Opis szczegółowy

Klasa zapisująca określone dane do pliku wynikowego

4.16.2 Dokumentacja funkcji składowych

4.16.2.1 ClearFile()

```
bool Save::ClearFile (  
    Data * DATA )
```

Publiczna metoda służąca do czyszczenia pliku przed zapisem danych

Parametry

<i>DATA</i>	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.16.2.2 SaveData()

```
bool Save::SaveData (
    Data * DATA )
```

Publiczna metoda zapisująca dane do pliku wynikowego

Parametry

<i>DATA</i>	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.16.2.3 SaveGraph()

```
bool Save::SaveGraph (
    Data * DATA )
```

Publiczna metoda zapisująca graf do pliku wynikowego

Parametry

<i>DATA</i>	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
-------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.16.2.4 SaveSolution()

```
bool Save::SaveSolution (
    Data * DATA )
```

Publiczna metoda zapisująca rozwiązanie do pliku wynikowego

Parametry

DATA	Wskaźnik na referencje do obiektu typu Data w którym przechowywane są wszystkie inne obiekty na których operuje program
------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Save.h](#)
- [Save.cpp](#)

4.17 Dokumentacja klasy Solution

```
#include <Solution.h>
```

Metody publiczne

- [Solution](#) ()
- [Solution](#) & [operator=](#) (const [Solution](#) &)
- bool [DeleteSolutionList](#) ()

Statyczne metody publiczne

- static bool [AddSolutionEnd](#) ([Solution](#) *[&](#), [Solution](#) *)
- static bool [AddSolutionByStart](#) ([Solution](#) *[&](#), [Solution](#) *)

Atrybuty publiczne

- [Primal](#) * [start](#)
Wskaźnik na element typu [primal](#) będący wierzchołkiem dla którego liczymy trasy do reszty wierzchołków.
- [Branch](#) * [branch](#)
Wskaźnik na element typu [Branch](#) zawierający wierzchołek końcowy i trasę do niego.
- [Solution](#) * [pNext](#)
Wskaźnik na następny element.
- [Solution](#) * [pPrev](#)
Wskaźnik na poprzedni element.

4.17.1 Opis szczegółowy

Klasa będąca odpowiedzialna za listę z rozwiązaniem

4.17.2 Dokumentacja konstruktora i destruktora

4.17.2.1 Solution()

```
Solution::Solution ( )
```

Konstruktor domyślny klasy [Solution](#)

4.17.3 Dokumentacja funkcji składowych

4.17.3.1 AddSolutionByStart()

```
bool Solution::AddSolutionByStart (
    Solution * & LS,
    Solution * S ) [static]
```

Publiczna metoda dodająca element typu [Solution](#) z uwzględnieniem numeru wierzchołka startowego

Parametry

LS	Wskaźnik na referencje do listy typu Solution
S	Wskaźnik na obiekt typu Solution

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.17.3.2 AddSolutionEnd()

```
bool Solution::AddSolutionEnd (
    Solution * & LS,
    Solution * S ) [static]
```

Publiczna metoda dodająca element typu [Solution](#) na koniec listy

Parametry

LS	Wskaźnik na referencje do listy typu Solution
S	Wskaźnik na obiekt typu Solution

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.17.3.3 DeleteSolutionList()

```
bool Solution::DeleteSolutionList ( )
```

Publiczna metoda do usuwania listy typu [Solution](#)

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.17.3.4 operator=()

```
Solution & Solution::operator= (
    const Solution & S )
```

Przeciążony operator przypisania

Parametry

S	Referencja do stałego obiektu typu Solution
---	---

Zwraca

Referencja do elementu typu [Solution](#) pozwalająca na kaskadowe wywołanie

Dokumentacja dla tej klasy została wygenerowana z plików:

- [Solution.h](#)
- [Solution.cpp](#)

4.18 Dokumentacja klasy User

```
#include <User.h>
```

Metody publiczne

- bool `Calibration` (`Data` *`&`, `std::string` `p`[])
- bool `Ask` (`Data` *`&`)

4.18.1 Opis szczegółowy

Klasa odpowiedzialna za obsługę komunikacji z użytkownikiem i pozyskaniem od niego niezbędnych informacji

4.18.2 Dokumentacja funkcji składowych

4.18.2.1 `Ask()`

```
bool User::Ask (  
    Data *& DATA )
```

Publiczna metoda do komunikacji z użytkownikiem w której przekazywane są preferencje na temat tego jak ma działać program

Parametry

<code>DATA</code>	Wskaźnik na referencje do obiektu typu <code>Data</code> w który przechowywane są wszystkie inne obiekty na których operuje program
-------------------	---

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

4.18.2.2 `Calibration()`

```
bool User::Calibration (  
    Data *& DATA,  
    std::string p[] )
```

Publiczna metoda przygotowująca podstawowe dane na podstawie informacji uzyskanych z konsoli

Parametry

<code>DATA</code>	Wskaźnik na referencje do obiektu typu <code>Data</code> w który przechowywane są wszystkie inne obiekty na których operuje program
<code>p</code>	Tablica typu <code>string</code> przechowująca ścieżki do plików wejścia/wyjścia

Zwraca

Zwraca informację o poprawności wykonania metody i wystąpieniu ewentualnego błędu

Dokumentacja dla tej klasy została wygenerowana z plików:

- [User.h](#)
- [User.cpp](#)

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku Algorithm.cpp

```
#include "Algorithm.h"
```

5.1.1 Opis szczegółowy

5.2 Dokumentacja pliku Algorithm.h

```
#include "Data.h"
```

Komponenty

- class [Algorithm](#)

5.2.1 Opis szczegółowy

5.3 Dokumentacja pliku Borderer.cpp

```
#include "Borderer.h"
```

5.3.1 Opis szczegółowy

5.4 Dokumentacja pliku Borderer.h

```
#include <iostream>
#include "Primal.h"
#include "DataTops.h"
```

Komponenty

- class [Borderer](#)

5.4.1 Opis szczegółowy

5.5 Dokumentacja pliku Branch.cpp

```
#include "Branch.h"
```

5.5.1 Opis szczegółowy

5.6 Dokumentacja pliku Branch.h

```
#include "Borderer.h"
#include "Primal.h"
```

Komponenty

- class [Branch](#)

5.6.1 Opis szczegółowy

5.7 Dokumentacja pliku Create.cpp

```
#include "Create.h"  
#include <iostream>  
#include <fstream>
```

5.7.1 Opis szczegółowy

5.8 Dokumentacja pliku Create.h

```
#include "Data.h"
```

Komponenty

- class [Create](#)

5.8.1 Opis szczegółowy

5.9 Dokumentacja pliku Data.cpp

```
#include "Data.h"
```

5.9.1 Opis szczegółowy

5.10 Dokumentacja pliku Data.h

```
#include <Windows.h>
#include "Operators.h"
#include "Signs.h"
```

Komponenty

- class [Data](#)

5.10.1 Opis szczegółowy

5.11 Dokumentacja pliku DataTops.cpp

```
#include "DataTops.h"
```

5.11.1 Opis szczegółowy

5.12 Dokumentacja pliku DataTops.h

```
#include "Primal.h"
```

Komponenty

- class [DataTops](#)

5.12.1 Opis szczegółowy

5.13 Dokumentacja pliku Graph.cpp

```
#include "Graph.h"
```

5.13.1 Opis szczegółowy

5.14 Dokumentacja pliku Graph.h

```
#include <iostream>
#include "Borderer.h"
#include "Primal.h"
```

Komponenty

- class [Graph](#)

5.14.1 Opis szczegółowy

5.15 Dokumentacja pliku Management.cpp

```
#include "Management.h"
```

5.15.1 Opis szczegółowy

5.16 Dokumentacja pliku Management.h

```
#include <iostream>
#include <string>
#include <fstream>
#include "Signs.h"
#include "Operators.h"
#include "Parameters.h"
#include "Data.h"
#include "Primal.h"
#include "Borderer.h"
#include "Predecessor.h"
#include "DataTops.h"
#include "Graph.h"
#include "Solution.h"
#include "User.h"
#include "Create.h"
#include "Algorithm.h"
#include "Save.h"
#include "PrintGraph.h"
#include "PrintData.h"
#include "PrintSolution.h"
```

Komponenty

- class [Management](#)

5.16.1 Opis szczegółowy

5.17 Dokumentacja pliku Operators.cpp

```
#include "Operators.h"
```

Funkcje

- std::ostream & [operator<<](#) (std::ostream &print, [Graph](#) *&G)
- std::ostream & [operator<<](#) (std::ostream &print, [Borderer](#) *&B)
- std::ostream & [operator<<](#) (std::ostream &print, [DataTops](#) *&D)
- std::ostream & [operator<<](#) (std::ostream &print, [Primal](#) *&P)
- std::ostream & [operator<<](#) (std::ostream &print, [Branch](#) *&B)
- std::ostream & [operator<<](#) (std::ostream &print, [Solution](#) *&S)

5.17.1 Opis szczegółowy

5.17.2 Dokumentacja funkcji

5.17.2.1 operator<<() [1/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Borderer *& B )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [Borderer](#)

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu Borderer
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.17.2.2 operator<<() [2/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Branch *& B )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [Branch](#)

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu Branch
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.17.2.3 operator<<() [3/6]

```
std::ostream& operator<< (
    std::ostream & print,
    DataTops *& D )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [DataTops](#) zawierającego dane wejściowe

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu DataTops
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.17.2.4 operator<<() [4/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Graph *& G )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania Grafu na ekranie

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy z grafem
--------------	--

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.17.2.5 operator<<() [5/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Primal *& P )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [Primal](#)

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu Primal
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.17.2.6 operator<<() [6/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Solution *& S )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania rozwiązania

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy zawierającej rozwiązanie
--------------	--

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.18 Dokumentacja pliku Operators.h

```
#include <iostream>
#include "Signs.h"
#include "Parameters.h"
#include "Primal.h"
#include "Borderer.h"
#include "Branch.h"
#include "Predecessor.h"
#include "DataTops.h"
#include "Graph.h"
#include "Solution.h"
```

Funkcje

- std::ostream & [operator<<](#) (std::ostream &, [Graph](#) *&)
- std::ostream & [operator<<](#) (std::ostream &, [Borderer](#) *&)
- std::ostream & [operator<<](#) (std::ostream &, [DataTops](#) *&)
- std::ostream & [operator<<](#) (std::ostream &, [Primal](#) *&)
- std::ostream & [operator<<](#) (std::ostream &, [Branch](#) *&)
- std::ostream & [operator<<](#) (std::ostream &, [Solution](#) *&)

5.18.1 Opis szczegółowy

5.18.2 Dokumentacja funkcji

5.18.2.1 operator<<() [1/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Borderer *& B )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [Borderer](#)

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu Borderer
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.18.2.2 operator<<() [2/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Branch *& B )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [Branch](#)

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu Branch
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.18.2.3 operator<<() [3/6]

```
std::ostream& operator<< (
    std::ostream & print,
    DataTops *& D )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [DataTops](#) zawierającego dane wejściowe

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu DataTops
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.18.2.4 operator<<() [4/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Graph *& G )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania Grafu na ekranie

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy z grafem
--------------	--

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.18.2.5 operator<<() [5/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Primal *& P )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania obiektu typu [Primal](#)

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy typu Primal
--------------	---

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.18.2.6 operator<<() [6/6]

```
std::ostream& operator<< (
    std::ostream & print,
    Solution *& S )
```

Przeciążony operator strumienia wykorzystywany do wyświetlania rozwiązania

Parametry

<i>print</i>	Referencja na wskaźnik wskazujący na pierwszy element listy zawierającej rozwiązanie
--------------	--

Zwraca

Zwraca referencje na strumień co pozwala na kaskadowe wywoływanie

5.19 Dokumentacja pliku Parameters.cpp

```
#include "Parameters.h"
```

5.19.1 Opis szczegółowy

5.20 Dokumentacja pliku Parameters.h

```
#include <cstdbool>
#include <string>
```

Komponenty

- class [Parameters](#)

5.20.1 Opis szczegółowy

5.21 Dokumentacja pliku Predecessor.cpp

```
#include "Predecessor.h"
```

5.21.1 Opis szczegółowy

5.22 Dokumentacja pliku Predecessor.h

```
#include <iostream>
#include "Borderer.h"
#include "Graph.h"
```

Komponenty

- class [Predecessor](#)

5.22.1 Opis szczegółowy

5.23 Dokumentacja pliku Primal.cpp

```
#include "Primal.h"
```

5.23.1 Opis szczegółowy

5.24 Dokumentacja pliku Primal.h

```
#include <iostream>
```

Komponenty

- class [Primal](#)

5.24.1 Opis szczegółowy

5.25 Dokumentacja pliku Print.h

```
#include "Data.h"
```

Komponenty

- class [Print](#)

5.25.1 Opis szczegółowy

5.26 Dokumentacja pliku PrintData.cpp

```
#include "PrintData.h"
```

5.26.1 Opis szczegółowy

5.27 Dokumentacja pliku PrintData.h

```
#include "Print.h"
```

Komponenty

- class [PrintData](#)

5.27.1 Opis szczegółowy

5.28 Dokumentacja pliku PrintGraph.cpp

```
#include "PrintGraph.h"
```

5.28.1 Opis szczegółowy

5.29 Dokumentacja pliku PrintGraph.h

```
#include "Print.h"
```

Komponenty

- class [PrintGraph](#)

5.29.1 Opis szczegółowy

5.30 Dokumentacja pliku PrintSolution.cpp

```
#include "PrintSolution.h"
```

5.30.1 Opis szczegółowy

5.31 Dokumentacja pliku PrintSolution.h

```
#include "Print.h"
```

Komponenty

- class [PrintSolution](#)

5.31.1 Opis szczegółowy

5.32 Dokumentacja pliku Save.cpp

```
#include "Save.h"
```

5.32.1 Opis szczegółowy

5.33 Dokumentacja pliku Save.h

```
#include <iostream>
#include <fstream>
#include "Data.h"
```

Komponenty

- class [Save](#)

5.33.1 Opis szczegółowy

5.34 Dokumentacja pliku Signs.h

```
#include <string>
```

5.34.1 Opis szczegółowy

5.35 Dokumentacja pliku Solution.cpp

```
#include "Solution.h"
```

5.35.1 Opis szczegółowy

5.36 Dokumentacja pliku Solution.h

```
#include "Parameters.h"  
#include "Primal.h"  
#include "Borderer.h"  
#include "Branch.h"  
#include "Predecessor.h"  
#include "DataTops.h"  
#include "Graph.h"
```

Komponenty

- class [Solution](#)

5.36.1 Opis szczegółowy

5.37 Dokumentacja pliku Source.cpp

```
#include <iostream>  
#include <string>  
#include <Windows.h>  
#include "Management.h"
```

Funkcje

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

5.37.1 Opis szczegółowy

5.38 Dokumentacja pliku User.cpp

```
#include "User.h"
```

5.38.1 Opis szczegółowy

5.39 Dokumentacja pliku User.h

```
#include "Data.h"
```

Komponenty

- class [User](#)

5.39.1 Opis szczegółowy

Indeks

- ~Data
 - Data, [17](#)
- AddBordererByNumber
 - Borderer, [10](#)
- AddBordererEnd
 - Borderer, [10](#)
- AddBranchByEnd
 - Branch, [12](#)
- AddBranchEnd
 - Branch, [13](#)
- AddDataTopsByTop
 - DataTops, [18](#)
- AddDataTopsEnd
 - DataTops, [19](#)
- AddGraphByTop
 - Graph, [21](#)
- AddGraphEnd
 - Graph, [22](#)
- AddPredecessorByNumber
 - Predecessor, [27](#)
- AddPredecessorEnd
 - Predecessor, [27](#)
- AddPrimalByNumber
 - Primal, [31](#)
- AddPrimalEnd
 - Primal, [32](#)
- AddPrimalFront
 - Primal, [32](#)
- AddSolutionByStart
 - Solution, [44](#)
- AddSolutionEnd
 - Solution, [44](#)
- Algorithm, [7](#)
 - CreateSolution, [7](#)
 - Dijkstra, [8](#)
 - PrepareSolution, [8](#)
- Algorithm.cpp, [49](#)
- Algorithm.h, [49](#)
- Ask
 - User, [46](#)
- Borderer, [9](#)
 - AddBordererByNumber, [10](#)
 - AddBordererEnd, [10](#)
 - Borderer, [9](#)
 - DeleteBordererList, [10](#)
 - FindBordererByNumber, [11](#)
 - operator=, [11](#)
- Borderer.cpp, [49](#)
- Borderer.h, [50](#)
- Branch, [11](#)
 - AddBranchByEnd, [12](#)
 - AddBranchEnd, [13](#)
 - Branch, [12](#)
 - DeleteBranchList, [13](#)
 - operator=, [13](#)
- Branch.cpp, [50](#)
- Branch.h, [50](#)
- Calibration
 - User, [46](#)
- CheckPrimalByNumber
 - Primal, [32](#)
- ClearFile
 - Save, [41](#)
- CompletePredecessor
 - Predecessor, [28](#)
- Coordinate
 - Management, [24](#)
- CopyPrimal
 - Primal, [33](#)
- CopyPrimalList
 - Primal, [33](#)
- Create, [14](#)
 - CreateData, [14](#)
 - CreateGraph, [15](#)
 - LoadData, [15](#)
 - LoadGraph, [15](#)
- Create.cpp, [51](#)
- Create.h, [51](#)
- CreateData
 - Create, [14](#)
- CreateGraph
 - Create, [15](#)
- CreatePrimalList
 - DataTops, [19](#)
- CreateSolution
 - Algorithm, [7](#)
- Data, [16](#)
 - ~Data, [17](#)
 - Data, [17](#)
- Data.cpp, [51](#)
- Data.h, [52](#)
- DataTops, [18](#)
 - AddDataTopsByTop, [18](#)
 - AddDataTopsEnd, [19](#)
 - CreatePrimalList, [19](#)
 - DataTops, [18](#)

- DeleteDataTopsList, 19
 - operator=, 20
 - pNext, 20
- DataTops.cpp, 52
- DataTops.h, 52
- DeleteBordererList
 - Borderer, 10
- DeleteBranchList
 - Branch, 13
- DeleteDataTopsList
 - DataTops, 19
- DeleteGraphList
 - Graph, 22
- DeletePredecessorList
 - Predecessor, 28
- DeletePrimalByNumber
 - Primal, 34
- DeletePrimalEnd
 - Primal, 34
- DeletePrimalList
 - Primal, 34
- DeleteSolutionList
 - Solution, 45
- Dijkstra
 - Algorithm, 8
- FindBordererByNumber
 - Borderer, 11
- FindPredecessorByLowerCost
 - Predecessor, 28
- FindPredecessorByNumber
 - Predecessor, 29
- FindPrimalByNumber
 - Primal, 34
- Graph, 20
 - AddGraphByTop, 21
 - AddGraphEnd, 22
 - DeleteGraphList, 22
 - Graph, 21
 - operator=, 22
- Graph.cpp, 53
- Graph.h, 53
- LoadData
 - Create, 15
- LoadGraph
 - Create, 15
- Management, 23
 - Coordinate, 24
 - Management, 24
- Management.cpp, 53
- Management.h, 54
- operator<<
 - Operators.cpp, 55–57
 - Operators.h, 58, 59
- operator=
 - Borderer, 11
 - Branch, 13
 - DataTops, 20
 - Graph, 22
 - Predecessor, 29
 - Primal, 35
 - Solution, 45
- Operators.cpp, 54
 - operator<<, 55–57
- Operators.h, 57
 - operator<<, 58, 59
- Parameters, 25
 - Parameters, 25
- Parameters.cpp, 60
- Parameters.h, 60
- pNext
 - DataTops, 20
- Predecessor, 26
 - AddPredecessorByNumber, 27
 - AddPredecessorEnd, 27
 - CompletePredecessor, 28
 - DeletePredecessorList, 28
 - FindPredecessorByLowerCost, 28
 - FindPredecessorByNumber, 29
 - operator=, 29
 - Predecessor, 27
 - PreparePredecessor, 29
- Predecessor.cpp, 60
- Predecessor.h, 61
- PreparePredecessor
 - Predecessor, 29
- PrepareSolution
 - Algorithm, 8
- Primal, 30
 - AddPrimalByNumber, 31
 - AddPrimalEnd, 32
 - AddPrimalFront, 32
 - CheckPrimalByNumber, 32
 - CopyPrimal, 33
 - CopyPrimalList, 33
 - DeletePrimalByNumber, 34
 - DeletePrimalEnd, 34
 - DeletePrimalList, 34
 - FindPrimalByNumber, 34
 - operator=, 35
 - Primal, 31
- Primal.cpp, 61
- Primal.h, 61
- Print, 35
- Print.h, 62
- PrintData, 36
 - PrintData, 36, 37
 - PrintDijkstra, 37
- PrintData.cpp, 62
- PrintData.h, 62
- PrintDijkstra
 - PrintData, 37
 - PrintGraph, 39

- PrintSolution, [40](#)
- PrintGraph, [38](#)
 - PrintDijkstra, [39](#)
 - PrintGraph, [38](#)
- PrintGraph.cpp, [63](#)
- PrintGraph.h, [63](#)
- PrintSolution, [39](#)
 - PrintDijkstra, [40](#)
 - PrintSolution, [40](#)
- PrintSolution.cpp, [63](#)
- PrintSolution.h, [63](#)
- Save, [41](#)
 - ClearFile, [41](#)
 - SaveData, [42](#)
 - SaveGraph, [42](#)
 - SaveSolution, [42](#)
- Save.cpp, [64](#)
- Save.h, [64](#)
- SaveData
 - Save, [42](#)
- SaveGraph
 - Save, [42](#)
- SaveSolution
 - Save, [42](#)
- Signs.h, [64](#)
- Solution, [43](#)
 - AddSolutionByStart, [44](#)
 - AddSolutionEnd, [44](#)
 - DeleteSolutionList, [45](#)
 - operator=, [45](#)
 - Solution, [44](#)
- Solution.cpp, [65](#)
- Solution.h, [65](#)
- Source.cpp, [65](#)
- User, [45](#)
 - Ask, [46](#)
 - Calibration, [46](#)
- User.cpp, [66](#)
- User.h, [66](#)