Evolution

Создано системой Doxygen 1.8.14

Оглавление

Глава 1

Титульная страница

Моделирование Эволюции

Версия

0.1.0

Автор

Мария Соловьева и Анастасия Важенина

Глава 2

Иерархический список классов

2.1 Иерархия классов

Иерархия классов.

Глава 3

Алфавитный указатель классов

3.1 Классы

Классы с их кратким описанием.

Глава 4

Список файлов

4.1 Файлы

Полный список файлов.

Глава 5

Классы

5.1 Класс Brain

Класс, описывающий мозг организма

#include <Brain.hpp>

Открытые члены

- Brain (const size_t &inInputs=10, const size_t &inOutputs=7, const size_t &inNumOfHidden↩ Layers=2, const size_t &inNumOfNeuronsInHiddenLayers=10)

  Конструктор создающий нейронную сеть с заданными параметрами
- Brain (const Json &json)

  Конструктор создающий нейронную сеть из данных формата Json.
- ~Brain ()=default

  Деструктор по умолчанию
- Brain & operator= (const Brain &brain)

  Конструктор копирования
- void Train ()

  Функция изменения(тренировки) мозга(нейронной сети) организма
- std::vector< Neuron * > GetLayer (size_t n) const

  Функция для получения слоя нейронной сети по номеру
- size_t Size () const

  Функция для получения количества слоев в нейронной сети
- std::vector< Neuron * > GetOutputLayer () const

  Функция по получению выходного слоя нейронной сети
- std::vector< Neuron * > GetInputLayer () const

  Функция по получению входного слоя нейронной сети
- size_t GetInputs () const

  Функция по получению количества входных нейронов
- void ResetWeights () const

  Функция, обнуляющая все веса нейронной сети
- const std::vector< double > CreateVectorInput (const std::vector< Hexagon *> &Surrounding↩ Objects) const

  Функция, преобразующая данные об окружающих объектах
- double Think (const std::vector< Hexagon *> &surroundingObjects3) const

Функция, высчитывающая степень уверенности нейронной сети

- Hexagon * GetSolution (const std::vector< Hexagon *> &surroundingObjects6) const

  Функция, принимающая решение о самом выгодном направлении перемещения

- void UpdateStateOfLife (double newST)

  Функция, обновляющая состояние жизни организма

- const Json getJson () const

  Функция по преобразованию мозга(нейронной сети) в формат Json.

5.1.1 Подробное описание

Класс, описывающий мозг организма

См. определение в файле Brain.hpp строка 22

5.1.2 Конструктор(ы)

5.1.2.1 Brain() [1/2]

```
Brain::Brain (
            const size_t & inInputs = 10,
            const size_t & inOutputs = 7,
            const size_t & inNumOfHiddenLayers = 2,
            const size_t & inNumOfNeuronsInHiddenLayers = 10 )
```

Конструктор создающий нейронную сеть с заданными параметрами

Аргументы

inInputs	количество входных нейронов
inOutputs	количество выходных нейронов
inNumOfHiddenLayers	количество скрытых слоев
inNumOfNeuronsInHiddenLayers	количество нейронов в скрытых слоях

См. определение в файле Brain.cpp строка 5

5.1.2.2 Brain() [2/2]

```
Brain::Brain (
            const Json & json )
```

Конструктор создающий нейронную сеть из данных формата Json.

Аргументы

json	данные, которые нужно преобразовать в объект класса Brain

См. определение в файле Brain.cpp строка 51

5.1.2.3 ~Brain()

Brain::~Brain ( ) [default]

Деструктор по умолчанию

5.1.3 Методы

5.1.3.1 CreateVectorInput()

const std::vector< double > Brain::CreateVectorInput (
const std::vector< Hexagon *> & SurroundingObjects) const

Функция, преобразующая данные об окружающих объектах

Аргументы

SurroundingObjects	вектор указателей на объекты, окружающие организм

Возвращает

вектор входных значений для нейронной сети

См. определение в файле Brain.cpp строка 141

5.1.3.2 GetInputLayer()

std::vector< Neuron * > Brain::GetInputLayer ( ) const

Функция по получению входного слоя нейронной сети

Возвращает

входной слой нейронной сети

См. определение в файле Brain.cpp строка 126

5.1.3.3 GetInputs()

size_t Brain::GetInputs ( ) const

Функция по получению количества входных нейронов

Возвращает

количество входных нейронов

См. определение в файле Brain.cpp строка 131

5.1.3.4 getJson()

const Json Brain::getJson ( ) const

Функция по преобразованию мозга(нейронной сети) в формат Json.

Возвращает

мозг(нейронная сеть), преобразованная в формат Json

См. определение в файле Brain.cpp строка 253

5.1.3.5 GetLayer()

std::vector< Neuron * > Brain::GetLayer (
size_t n) const

Функция для получения слоя нейронной сети по номеру

Аргументы

n	номер слоя в нейронной сети

Возвращает

слой нейронной сети

См. определение в файле Brain.cpp строка 111

5.1.3.6 GetOutputLayer()

std::vector< Neuron * > Brain::GetOutputLayer ( ) const

Функция по получению выходного слоя нейронной сети

Возвращает

выходной слой нейронной сети

См. определение в файле Brain.cpp строка 121

5.1.3.7 GetSolution()

Hexagon * Brain::GetSolution (
const std::vector< Hexagon *> & surroundingObjects6) const

Функция, принимающая решение о самом выгодном направлении перемещения

Аргументы

surroundingObjects6	вектор указателей на объекты, окружающих организм

Возвращает

указатель на объект, в направлении которого следует перемещаться

См. определение в файле Brain.cpp строка 200

5.1.3.8 operator=()

Brain & Brain::operator= (
const Brain & brain)

Конструктор копирования

Аргументы

brain	мозг, который нужно скопировать

См. определение в файле Brain.cpp строка 97

5.1.3.9 ResetWeights()

void Brain::ResetWeights ( ) const

Функция, обнуляющая все веса нейронной сети

См. определение в файле Brain.cpp строка 237

5.1.3.10 Size()

size_t Brain::Size ( ) const

Функция для получения количества слоев в нейронной сети

Возвращает

количество слоёв в нейронной сети

См. определение в файле Brain.cpp строка 116

5.1.3.11 Think()

double Brain::Think (
const std::vector< Hexagon *> & surroundingObjects3) const

Функция, высчитывающая степень уверенности нейронной сети

Аргументы

surroundingObjects3	вектор указателей на 3 объекта, попадающие в его поле зрения

Возвращает

степень уверенности нейронной сети(стоит ли перемещаться в данном напрвлении)

См. определение в файле Brain.cpp строка 158

5.1.3.12 Train()

void Brain::Train ( )

Функция изменения(тренировки) мозга(нейронной сети) организма

См. определение в файле Brain.cpp строка 136

5.1.3.13 UpdateStateOfLife()

```
void Brain::UpdateStateOfLife (
                double newST )
```

Функция, обновляющая состояние жизни организма

Аргументы

newST	новое состояние жизни организма

См. определение в файле Brain.cpp строка 248

Объявления и описания членов классов находятся в файлах:

- Brain.hpp
- Brain.cpp

5.2 Класс Button

Класс кнопки

#include <Button.hpp>

Открытые члены

- Button ()=default

  Конструктор по умочанию
- ~Button ()=default

  Деструктор по умолчанию
- Button (const float x, const float y, const float rotation)

  Конструктор, создающий кнопку с заданными координатами и поворотом
- double GetX ()

  Функция для получения координаты кнопки по оси Ox.
- double GetY ()

  Функция для получения координаты кнопки по оси Oy.
- sf::Color GetColor ()

  Функция для получения цвета кнопки
- sf::CircleShape GetButton ()

  Функция для получения кнопки-треугольника
- void SetColor (sf::Color newColor)

  Функция для получения цвета кнопки
- void SetX (double newX)

  Функция для получения цвета кнопки
- void SetY (double newY)

  Функция для получения цвета кнопки
- void Print (sf::RenderWindow *window)

  Функция для получения цвета кнопки

5.2.1 Подробное описание

Класс кнопки

См. определение в файле Button.hpp строка 11

5.2.2 Конструктор(ы)

5.2.2.1 Button() [1/2]

```
Button::Button ( )  [default]
```

Конструктор по умочанию

5.2.2.2 ~Button()

```
Button::~Button ( )  [default]
```

Деструктор по умолчанию

5.2.2.3 Button() [2/2]

```
Button::Button (
            const float x,
            const float y,
            const float rotation )
```

Конструктор, создающий кнопку с заданными координатами и поворотом

Аргументы

x	расположения кнопки в пикселях по оси Ox на окне
y	расположения кнопки в пикселях по оси Oy на окне
rotation	угол поворота

См. определение в файле Button.cpp строка 3

5.2.3 Методы

5.2.3.1 GetButton()

sf::CircleShape Button::GetButton ( )

Функция для получения кнопки-треугольника

Возвращает

кнопка-треугольник

См. определение в файле Button.cpp строка 25

5.2.3.2 GetColor()

sf::Color Button::GetColor ( )

Функция для получения цвета кнопки

Возвращает

цвет кнопки

См. определение в файле Button.cpp строка 30

5.2.3.3 GetX()

double Button::GetX ( )

Функция для получения координаты кнопки по оси Ox.

Возвращает

координаты кнопки по оси Ox

См. определение в файле Button.cpp строка 15

5.2.3.4 GetY()

double Button::GetY ( )

Функция для получения координаты кнопки по оси Oy.

Возвращает

координаты кнопки по оси Oy

См. определение в файле Button.cpp строка 20

5.2.3.5 Print()

```
void Button::Print (
            sf::RenderWindow * window )
```

Функция для получения цвета кнопки

Возвращает

цвет кнопки

См. определение в файле Button.cpp строка 50

5.2.3.6 SetColor()

```
void Button::SetColor (
            sf::Color newColor )
```

Функция для получения цвета кнопки

Возвращает

цвет кнопки

См. определение в файле Button.cpp строка 35

5.2.3.7 SetX()

```
void Button::SetX (
            double newX )
```

Функция для получения цвета кнопки

Возвращает

цвет кнопки

См. определение в файле Button.cpp строка 40

5.2.3.8 SetY()

```
void Button::SetY (
            double newY )
```

Функция для получения цвета кнопки

Возвращает

цвет кнопки

См. определение в файле Button.cpp строка 45

Объявления и описания членов классов находятся в файлах:

- Button.hpp
- Button.cpp

5.3 Класс Evolution

Класс Эволюции

#include <Evolution.hpp>

Открытые члены

- Evolution ()
  Конструктор по умочанию
- ~Evolution ()=default
  Деструктор по умолчанию
- void run ()
  Функция запуска эволюции
- void Statistics ()
  Функция сбора статистики
- void Menu ()
  Функция, показа меню

5.3.1 Подробное описание

Класс Эволюции

См. определение в файле Evolution.hpp строка 20

5.3.2 Конструктор(ы)

5.3.2.1 Evolution()

Evolution::Evolution ( )

Конструктор по умочанию

См. определение в файле Evolution.cpp строка 4

5.3.2.2 ~Evolution()

Evolution::~Evolution ( ) [default]

Деструктор по умолчанию

5.3.3 Методы

5.3.3.1 Menu()

void Evolution::Menu ( )

Функция, показа меню

См. определение в файле Evolution.cpp строка 29

5.3.3.2 run()

void Evolution::run ( )

Функция запуска эволюции

См. определение в файле Evolution.cpp строка 289

5.3.3.3 Statistics()

void Evolution::Statistics ( )

Функция сбора статистики

См. определение в файле Evolution.cpp строка 11

Объявления и описания членов классов находятся в файлах:

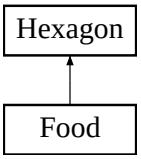
- Evolution.hpp
- Evolution.cpp

5.4 Класс Food

Класс Еды

#include <Food.hpp>

Граф наследования:Food:



Открытые члены

- Food (const double xNew, const double yNew, const size_t CellStrNew, const size_t CellColNew, double Medicine=rand() % 5)

  Конструктор, создающий гекс-еды с заданными координатами
- ~Food () override=default

  Деструктор по умолчанию
- void Print (sf::RenderWindow *window) const override

  Функция отрисовки гекса-еды

Дополнительные унаследованные члены

5.4.1 Подробное описание

Класс Еды

См. определение в файле Food.hpp строка 11

5.4.2 Конструктор(ы)

5.4.2.1 Food()

```
Food::Food (
            const double xNew,
            const double yNew,
            const size_t CellStrNew,
            const size_t CellColNew,
            double Medicine = rand() % 5 )   [inline]
```

Конструктор, создающий гекс-еды с заданными координатами

Аргументы

xNew	расположение гекса-еды в пикселях по оси Ox на окне
yNew	расположение гекса-еды в пикселях по оси Oy на окне
CellStrNew	номер строки в матрице карты
CellColNew	номер столбца в матрице карты
Medicine	степнь ядовитости еды

См. определение в файле Food.hpp строка 21

5.4.2.2 ~Food()

Food::~Food ( ) [override], [default]

Деструктор по умолчанию

5.4.3 Методы

5.4.3.1 Print()

void Food::Print (
sf::RenderWindow * window) const [inline], [override], [virtual]

Функция отрисовки гекса-еды

Аргументы

window	окно, в котором нужно отрисовать гекс-еду

Переопределяет метод предка Hexagon.

См. определение в файле Food.hpp строка 30

Объявления и описания членов класса находятся в файле:

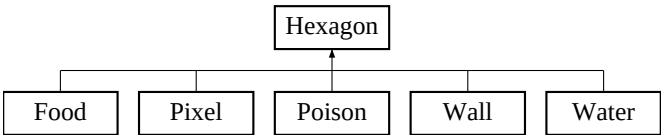
- Food.hpp

5.5 Класс Hexagon

Родительский класс, описывающий шестиугольник(гекс)

#include <Hexagon.hpp>

Граф наследования:Hexagon:



Открытые типы

- enum Type : int {
  FOOD = 1, WATER, POISON, PIXEL,
  WALL }

Открытые члены

- Hexagon ()=default

  Конструктор по умочанию
- Hexagon (const Type, double xInPixels, double yInPixels, size_t xInCells, size_t yInCells)

  Конструктор, создающий гекс определенного типа с заданными координатами
- Hexagon & operator= (const Hexagon *hex)

  Конструктор копирования
- virtual ~Hexagon ()=default

  Деструктор по умолчанию
- double GetX () const

  Функция для получения расположения гекса в пикселях по оси Ox на окне
- double GetY () const

  Функция для получения расположения гекса в пикселях по оси Oy на окне
- size_t GetCellStr () const

  Функция для получения номера строки в матрице карты, на которой расположен гекс
- size_t GetCellCol () const

  Функция для получения номера столбца в матрице карты, на котором расположен гекс
- Type GetType () const

  Функция для получения типа гекс(например, вода)
- double GetLifes () const

  Функция для получения количества жизней гекса
- double GetMedicine () const

  Функция для получения степени ядовитости гекса
- bool GetisHealfy () const

  Функция для получения состояния здоровья гекса
- void SetX (double x)

  Функция для установки расположения гекса в пикселях по оси Ox на окне
- void SetY (double y)

  Функция для установки расположения гекса в пикселях по оси Oy на окне
- void SetCellStr (size_t xInCells)

  Функция для установки номера строки в матрице карты, на которой расположен гекс
- void SetCellCol (size_t yInCells)

  Функция для установки номера столбца в матрице карты, на котором расположен гекс
- void SetType (Type &t)

  Функция для установки типа гекса(например, вода)

- void SetLifes (double l)

  Функция для установки колочества жизней гекса
- void SetMedicine (double m)

  Функция для установки степени ядовитости гекса
- void ResetMedicine ()

  Функция для обнуления степения ядовитости гекса
- bool IsAlive ()

  Функция для проверки жизнеспособности гекса
- virtual void Print (sf::RenderWindow *window) const

  Функция для отрисовки гекса

Защищенные данные

- double x

  расположение гекса в пикселях по оси Ox на окне
- double y

  расположения гекса в пикселях по оси Oy на окне
- size_t cellStr

  номер строки в матрице карты, на которой расположен гекс
- size_t cellCol

  номер столбца в матрице карты, на котором расположен гекс
- Type type

  тип гекса(например, вода)
- double lifes

  количество жизней гекса
- double medicine

  степень ядовитости гекса
- bool isHealfy = true

  состояние здоровья гекса

5.5.1 Подробное описание

Родительский класс, описывающий шестиугольник(гекс)

См. определение в файле Hexagon.hpp строка 16

5.5.2 Перечисления

5.5.2.1 Type

enum Hexagon::Type : int

Элементы перечислений

FOOD	
WATER	
POISON	
PIXEL	
WALL	

См. определение в файле Hexagon.hpp строка 20

5.5.3 Конструктор(ы)

5.5.3.1 Hexagon() [1/2]

Hexagon::Hexagon ( ) [default]

Конструктор по умочанию

5.5.3.2 Hexagon() [2/2]

```
Hexagon::Hexagon (
                const Type type1,
                double xInPixels,
                double yInPixels,
                size_t xInCells,
                size_t yInCells )
```

Конструктор, создающий гекс определенного типа с заданными координатами

Аргументы

Type	определитель типа гекса(например, вода)
xInPixels	расположение гекса в пикселях по оси Ox на окне
yInPixels	расположение гекса в пикселях по оси Oy на окне
xInCells	номер строки в матрице карты
yInCells	номер столбца в матрице карты

См. определение в файле Hexagon.cpp строка 5

5.5.3.3 ~Hexagon()

virtual Hexagon::~Hexagon ( ) [virtual], [default]

Деструктор по умолчанию

5.5.4 Методы

5.5.4.1 GetCellCol()

size_t Hexagon::GetCellCol ( ) const

Функция для получения номера столбца в матрице карты, на котором расположен гекс

Возвращает

номер столбца в матрице карты, на котором расположен гекс

См. определение в файле Hexagon.cpp строка 44

5.5.4.2 GetCellStr()

size_t Hexagon::GetCellStr ( ) const

Функция для получения номера строки в матрице карты, на которой расположен гекс

Возвращает

номер строки в матрице карты, на которой расположен гекс

См. определение в файле Hexagon.cpp строка 39

5.5.4.3 GetisHealfy()

bool Hexagon::GetisHealfy ( ) const

Функция для получения состояния здоровья гекса

Возвращает

состояние здоровья гекса

См. определение в файле Hexagon.cpp строка 64

5.5.4.4 GetLifes()

double Hexagon::GetLifes ( ) const

Функция для получения количества жизней гекса

Возвращает

количество жизней гекса

См. определение в файле Hexagon.cpp строка 54

5.5.4.5 GetMedicine()

double Hexagon::GetMedicine ( ) const

Функция для получения степени ядовитости гекса

Возвращает

степень ядовитости гекса

См. определение в файле Hexagon.cpp строка 59

5.5.4.6 GetType()

Hexagon::Type Hexagon::GetType ( ) const

Функция для получения типа гекс(например, вода)

Возвращает

тип гекса(например, вода)

См. определение в файле Hexagon.cpp строка 49

5.5.4.7 GetX()

double Hexagon::GetX ( ) const

Функция для получения расположения гекса в пикселях по оси Ox на окне

Возвращает

расположение гекса в пикселях по оси Ox на окне

См. определение в файле Hexagon.cpp строка 29

5.5.4.8 GetY()

double Hexagon::GetY ( ) const

Функция для получения расположения гекса в пикселях по оси Oy на окне

Возвращает

расположение гекса в пикселях по оси Oy на окне

См. определение в файле Hexagon.cpp строка 34

5.5.4.9 IsAlive()

bool Hexagon::IsAlive ( )

Функция для проверки жизнеспособности гекса

Возвращает

true - в случае, если гекс жив, иначе false

См. определение в файле Hexagon.cpp строка 112

5.5.4.10 operator=()

Hexagon & Hexagon::operator= (
const Hexagon * hex)

Конструктор копирования

Аргументы

hex	указатель на гекс, который нужно скопировать

См. определение в файле Hexagon.cpp строка 14

5.5.4.11 Print()

void Hexagon::Print (
sf::RenderWindow * window) const [virtual]

Функция для отрисовки гекса

Аргументы

window	указатель на окно, в котором требуется отрисовать гекс

Переопределяется в Pixel, Poison, Water и Food.

См. определение в файле Hexagon.cpp строка 117

5.5.4.12 ResetMedicine()

void Hexagon::ResetMedicine ( )

Функция для обнуления степения ядовитости гекса

См. определение в файле Hexagon.cpp строка 106

5.5.4.13 SetCellCol()

void Hexagon::SetCellCol (
size_t yInCells)

Функция для установки номера столбца в матрице карты, на котором расположен гекс

Аргументы

yInCells	номер столбца в матрице карты, на котором расположен гекс

См. определение в файле Hexagon.cpp строка 84

5.5.4.14 SetCellStr()

void Hexagon::SetCellStr (
size_t xInCells)

Функция для установки номера строки в матрице карты, на которой расположен гекс

Аргументы

xInCells	номер строки в матрице карты, на которой расположен гекс

См. определение в файле Hexagon.cpp строка 79

5.5.4.15 SetLifes()

void Hexagon::SetLifes (
double l)

Функция для установки колочества жизней гекса

Аргументы

l	количество жизней гекса

См. определение в файле Hexagon.cpp строка 95

5.5.4.16 SetMedicine()

void Hexagon::SetMedicine (
double m)

Функция для установки степени ядовитости гекса

Аргументы

m	степень ядовитости гекса

См. определение в файле Hexagon.cpp строка 100

5.5.4.17 SetType()

void Hexagon::SetType (
Type & t)

Функция для установки типа гекса(например, вода)

Аргументы

t	тип гекса(например, вода)

См. определение в файле Hexagon.cpp строка 90

5.5.4.18 SetX()

void Hexagon::SetX (
double x)

Функция для установки расположения гекса в пикселях по оси Ox на окне

Аргументы

x	расположение гекса в пикселях по оси Ox на окне

См. определение в файле Hexagon.cpp строка 69

5.5.4.19 SetY()

void Hexagon::SetY (
double y)

Функция для установки расположения гекса в пикселях по оси Oy на окне

Аргументы

y	расположение гекса в пикселях по оси Oy на окне

См. определение в файле Hexagon.cpp строка 74

5.5.5 Данные класса

5.5.5.1 cellCol

size_t Hexagon::cellCol [protected]

номер столбца в матрице карты, на котором расположен гекс

См. определение в файле Hexagon.hpp строка 104

5.5.5.2 cellStr

size_t Hexagon::cellStr [protected]

номер строки в матрице карты, на которой расположен гекс

См. определение в файле Hexagon.hpp строка 102

5.5.5.3 isHealfy

bool Hexagon::isHealfy = true [protected]

состояние здоровья гекса

См. определение в файле Hexagon.hpp строка 112

5.5.5.4 lifes

double Hexagon::lifes [protected]

количество жизней гекса

См. определение в файле Hexagon.hpp строка 108

5.5.5.5 medicine

double Hexagon::medicine [protected]

степень ядовитости гекса

См. определение в файле Hexagon.hpp строка 110

5.5.5.6 type

Type Hexagon::type [protected]

тип гекса(например, вода)

См. определение в файле Hexagon.hpp строка 106

5.5.5.7 x

double Hexagon::x [protected]

расположение гекса в пикселях по оси Ox на окне

См. определение в файле Hexagon.hpp строка 98

5.5.5.8 y

double Hexagon::y [protected]

расположения гекса в пикселях по оси Oy на окне

См. определение в файле Hexagon.hpp строка 100

Объявления и описания членов классов находятся в файлах:

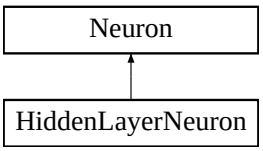
- Hexagon.hpp
- Hexagon.cpp

5.6 Класс HiddenLayerNeuron

Скрытый нейрон

#include <Neuron.hpp>

Граф наследования:HiddenLayerNeuron:



Открытые члены

- HiddenLayerNeuron (Neuron *inNeuron)
  Конструктор копирования

5.6.1 Подробное описание

Скрытый нейрон

См. определение в файле Neuron.hpp строка 96

5.6.2 Конструктор(ы)

5.6.2.1 HiddenLayerNeuron()

HiddenLayerNeuron::HiddenLayerNeuron (
Neuron * inNeuron) [inline], [explicit]

Конструктор копирования

Аргументы

inNeuron	указатель на нейрон, который следует скопировать

См. определение в файле Neuron.hpp строка 102

Объявления и описания членов класса находятся в файле:

- Neuron.hpp

5.7 Класс Keyboard

Класс Клавиатуры

#include <Keyboard.hpp>

Открытые члены

- void press (sf::Keyboard::Key key)
  Функция добавления ключа нажатой кнопки
- void release (sf::Keyboard::Key key)
  Функция отжатия кнопки
- bool isPressed (sf::Keyboard::Key key)
  Функция проверки : Нажата ли кнопка?

5.7.1 Подробное описание

Класс Клавиатуры

См. определение в файле Keyboard.hpp строка 8

5.7.2 Методы

5.7.2.1 isPressed()

bool Keyboard::isPressed (
sf::Keyboard::Key key)

Функция проверки : Нажата ли кнопка?

Аргументы

ключ	нажатой кнопки

Возвращает

true, если нажатаб false - иначе

См. определение в файле Keyboard.cpp строка 13

5.7.2.2 press()

void Keyboard::press (
sf::Keyboard::Key key)

Функция добавления ключа нажатой кнопки

Аргументы

ключ	нажатой кнопки

См. определение в файле Keyboard.cpp строка 3

5.7.2.3 release()

void Keyboard::release (
sf::Keyboard::Key key)

Функция отжатия кнопки

Аргументы

ключ	отжатой кнопки

См. определение в файле Keyboard.cpp строка 8

Объявления и описания членов классов находятся в файлах:

- Keyboard.hpp
- Keyboard.cpp

5.8 Класс Link

Класс Связи между нейронами

#include <Link.hpp>

Открытые члены

- Link ()
  Конструктор по умочанию
- Link (Neuron *neuronLinkedTo, double weightToNeuron=0)
  Класс Клавиатуры
- double GetWeight () const
  Функция получения веса связи
- void SetWeight (const double weight)
  Функция установки веса
- void SetNeuronLinkedTo (Neuron *neuronLinkedTo)
  Функция установки связи с нейроном
- Neuron * GetNeuronLinkedTo () const
  Функция получения нейрона, к которому присоединена связь
- void UpdateWeight (const double delta)
  Функция изменения веса связи

5.8.1 Подробное описание

Класс Связи между нейронами

См. определение в файле Link.hpp строка 7

5.8.2 Конструктор(ы)

5.8.2.1 Link() [1/2]

Link::Link ( )

Конструктор по умочанию

См. определение в файле Link.cpp строка 3

5.8.2.2 Link() [2/2]

Link::Link (
Neuron * neuronLinkedTo,
double weightToNeuron = 0) [explicit]

Класс Клавиатуры

Аргументы

neuronLinkedTo	
weightToNeuron	

См. определение в файле Link.cpp строка 8

5.8.3 Методы

5.8.3.1 GetNeuronLinkedTo()

Neuron * Link::GetNeuronLinkedTo ( ) const

Функция получения нейрона, к которому присоединена связь

Возвращает

нейрон, к которому присоединена связь

См. определение в файле Link.cpp строка 28

5.8.3.2 GetWeight()

double Link::GetWeight ( ) const

Функция получения веса связи

Возвращает

вес связи

См. определение в файле Link.cpp строка 13

5.8.3.3 SetNeuronLinkedTo()

void Link::SetNeuronLinkedTo (
Neuron * neuronLinkedTo)

Функция установки связи с нейроном

Аргументы

neuronLinkedTo	нейрон, с которым следует установить связь

См. определение в файле Link.cpp строка 23

5.8.3.4 SetWeight()

void Link::SetWeight (
const double weight)

Функция установки веса

Аргументы

weight	новый вес связи

См. определение в файле Link.cpp строка 18

5.8.3.5 UpdateWeight()

void Link::UpdateWeight (
const double delta)

Функция изменения веса связи

Аргументы

delta	значение, на которое изменяется занчение веса связи

См. определение в файле Link.cpp строка 33

Объявления и описания членов классов находятся в файлах:

- Link.hpp
- Link.cpp

5.9 Класс Map

#include <Map.hpp>

Открытые члены

- Map (size_t widthCells=94, size_t heightCells=60)

  Конструктор, создающий карту по заданным параметрам(по умолчанию количество строк - 60, столбцов - 94)
- Map (const std::string &path, int numberOfEvolution, size_t newWidth, size_t newHeight)

  Конструктор для загрузки карты из файла
- Map (const Map &mapToCopy)

  Конструктор копирования
- Map (Map &&mapToMove)

  Конструктор перемещения
- ~Map ()=default

  Деструктор по умочанию
- Map & operator= (const Map &mapToCopy)

  Оператор копирования
- Map & operator= (Map &&mapToMove)

  Оператор перемещения
- Row & operator[] (size_t index)

  Оператор обращения по индексу
- const Row & operator[] (size_t index) const

  Оператор обращения по индексу
- void Update ()

  Функция обновления карты
- void MultiplyPixels (int numberOfPixels)

  Функция запуска организмов на карту
- void CreateFood (int numberOfFood)

  Функция распространения еды по карте
- void SetPoison (int numberOfPoison)

  Функция распостранения яда по карте
- void RecreateMap (const std::vector< Pixel *> &organisms_)

  Функция пересоздания карты
- void ClonePixels (Map &map_, const std::vector< Pixel *> &organisms_)

  Функция клонирования организмов

- std::vector< Pixel * > Selection (const std::vector< Pixel *> &organisms_)

  Функция селекции
- unsigned int GetWidth () const

  Функция получения ширины карты
- unsigned int GetHeight () const

  Функция получения высоты карты
- size_t GetWidthInCells () const

  Функция получения количества столбцов в карте
- size_t GetHeightInCells () const

  Функция получения количества строк в карте
- std::vector< Pixel * > GetOrganisms () const

  Функция получения огранизмов, живущих на карте
- std::vector< Pixel * > GetStaticOrganisms () const

  Функция получения умерших огранизмов
- size_t GetNumberOfAliveOrganisms () const

  Функция получения числа живых организмов на карте
- unsigned int GetEvolutionNumber () const

  Функция получения номера эволюции
- void IncreaseEvolutionNumber ()

  Функция увеличения номера эволюции
- int GetTimeToSleep () const

  Функция получения задержки потока
- void IncreaseTimesToSleep (int delta)

  Функция увеличения задержки потока
- void DecreaseTimesToSleep (int delta)

  Функция уменьшения задержки потока
- Wall * GetWall () const

  Функция получения указателя на стену
- void SetOrganism (Pixel *pixel_)

  Функция установки организма на карту
- void Swap (Hexagon *hexagon1, Hexagon *hexagon2)

  Функция обмена координатами
- void SaveToFile () const

  Функция сохраниения в файл
- void UploadFromFile (int numberOfEvolution, size_t newWidth, size_t newHeight)

  Функция загрузки из файла
- void Print (sf::RenderWindow *window) const

  Функция отрисовки

5.9.1 Подробное описание

См. определение в файле Map.hpp строка 65

5.9.2 Конструктор(ы)

5.9.2.1 Map() [1/4]

```
Map::Map (
        size_t widthCells = 94,
        size_t heightCells = 60 )
```

Конструктор, создающий карту по заданным параметрам(по умолчанию количество строк - 60, столбцов - 94)

Аргументы

widthCells	ширина карты
heightCells	высота карты

См. определение в файле Map.cpp строка 7

5.9.2.2 Map() [2/4]

```
Map::Map (
        const std::string & path,
        int numberOfEvolution,
        size_t newWidth,
        size_t newHeight )
```

Конструктор для загрузки карты из файла

Аргументы

path	путь до файла, из которого нужно загрузить данные
numberOfEvolution	номер эволюции
newWidth	ширина карты
newHeight	высота карты

См. определение в файле Map.cpp строка 185

5.9.2.3 Map() [3/4]

```
Map::Map (
        const Map & mapToCopy )
```

Конструктор копирования

Аргументы

mapToCopy	карта, которую нужно скопировать

См. определение в файле Map.cpp строка 36

5.9.2.4 Map() [4/4]

Map::Map (
Map && mapToMove)

Конструктор перемещения

Аргументы

mapToMove	карта, которую следует переместить

См. определение в файле Map.cpp строка 46

5.9.2.5 ~Map()

Map::~Map ( ) [default]

Деструктор по умочанию

5.9.3 Методы

5.9.3.1 ClonePixels()

void Map::ClonePixels (
Map & map_,
const std::vector< Pixel *> & organisms_)

Функция клонирования организмов

Аргументы

map_	старая карта
organisms_	организмы, которых нужно размножить

См. определение в файле Map.cpp строка 151

5.9.3.2 CreateFood()

void Map::CreateFood (
int numberOfFood)

Функция распространения еды по карте

Аргументы

numberOfFood	количество еды, которое должно находиться на карте

См. определение в файле Map.cpp строка 76

5.9.3.3 DecreaseTimesToSleep()

void Map::DecreaseTimesToSleep (
int delta)

Функция уменьшения задержки потока

Аргументы

delta	значения уменьшения задержки потока

См. определение в файле Map.cpp строка 370

5.9.3.4 GetEvolutionNumber()

unsigned int Map::GetEvolutionNumber ( ) const

Функция получения номера эволюции

Возвращает

номер эволюции

См. определение в файле Map.cpp строка 350

5.9.3.5 GetHeight()

unsigned int Map::GetHeight ( ) const

Функция получения высоты карты

Возвращает

высота карты

См. определение в файле Map.cpp строка 320

5.9.3.6 GetHeightInCells()

size_t Map::GetHeightInCells ( ) const

Функция получения количества строк в карте

Возвращает

количество строк

См. определение в файле Map.cpp строка 330

5.9.3.7 GetNumberOfAliveOrganisms()

size_t Map::GetNumberOfAliveOrganisms ( ) const

Функция получения числа живых организмов на карте

Возвращает

число живых организмов на карте

См. определение в файле Map.cpp строка 345

5.9.3.8 GetOrganisms()

std::vector< Pixel * > Map::GetOrganisms ( ) const

Функция получения огранизмов, живущих на карте

Возвращает

вектор организмов, живущих на карте

См. определение в файле Map.cpp строка 335

5.9.3.9 GetStaticOrganisms()

std::vector< Pixel * > Map::GetStaticOrganisms ( ) const

Функция получения умерших огранизмов

Возвращает

вектор умерших организмов

См. определение в файле Map.cpp строка 340

5.9.3.10 GetTimeToSleep()

int Map::GetTimeToSleep ( ) const

Функция получения задержки потока

Возвращает

задержка потока

См. определение в файле Map.cpp строка 360

5.9.3.11 GetWall()

Wall * Map::GetWall ( ) const

Функция получения указателя на стену

Возвращает

указатель на стену

См. определение в файле Map.cpp строка 375

5.9.3.12 GetWidth()

unsigned int Map::GetWidth ( ) const

Функция получения ширины карты

Возвращает

ширина карты

См. определение в файле Map.cpp строка 315

5.9.3.13 GetWidthInCells()

size_t Map::GetWidthInCells ( ) const

Функция получения количества столбцов в карте

Возвращает

количество столбцов

См. определение в файле Map.cpp строка 325

5.9.3.14 IncreaseEvolutionNumber()

```
void Map::IncreaseEvolutionNumber ( )
```

Функция увеличения номера эволюции

См. определение в файле Map.cpp строка 355

5.9.3.15 IncreaseTimesToSleep()

```
void Map::IncreaseTimesToSleep (
            int delta )
```

Функция увеличения задержки потока

Аргументы

delta	значения увеличения задержки потока

См. определение в файле Map.cpp строка 365

5.9.3.16 MultiplyPixels()

```
void Map::MultiplyPixels (
            int numberOfPixels )
```

Функция запуска организмов на карту

Аргументы

numberOfPixels	количество организмов, которые должны жить на карте

См. определение в файле Map.cpp строка 55

5.9.3.17 operator=() [1/2]

```
Map & Map::operator= (
            const Map & mapToCopy )
```

Оператор копирования

Аргументы

mapToCopy	карта, которую следует скопировать

См. определение в файле Map.cpp строка 238

5.9.3.18 operator=() [2/2]

```
Map & Map::operator= (
            Map && mapToMove )
```

Оператор перемещения

Аргументы

mapToMove	карта, которую следует переместить

См. определение в файле Map.cpp строка 252

5.9.3.19 operator[]() [1/2]

```
Row & Map::operator[] (
            size_t index )
```

Оператор обращения по индексу

Аргументы

index	номер строки, к которой нужно обратиться

Возвращает

строка с индексом index

См. определение в файле Map.cpp строка 305

5.9.3.20 operator[]() [2/2]

```
const Row & Map::operator[] (
            size_t index ) const
```

Оператор обращения по индексу

Аргументы

index	номер строки, к которой нужно обратиться

Возвращает

строка с индексом index

См. определение в файле Map.cpp строка 310

5.9.3.21 Print()

```
void Map::Print (
            sf::RenderWindow * window ) const
```

Функция отрисовки

Аргументы

window	окно, в котором происходит отрисовка

См. определение в файле Map.cpp строка 436

5.9.3.22 RecreateMap()

```
void Map::RecreateMap (
            const std::vector< Pixel *> & organisms_ )
```

Функция пересоздания карты

Аргументы

organisms_	организмы, которые должны жить на новой карте

См. определение в файле Map.cpp строка 141

5.9.3.23 SaveToFile()

```
void Map::SaveToFile ( ) const
```

Функция сохраниения в файл

См. определение в файле Map.cpp строка 404

5.9.3.24 Selection()

std::vector< Pixel * > Map::Selection (
const std::vector< Pixel *> & organisms_)

Функция селекции

Аргументы

organisms↩ _	вектор организмов, из которых следует осуществить выборку

См. определение в файле Map.cpp строка 122

5.9.3.25 SetOrganism()

void Map::SetOrganism (
Pixel * pixel_)

Функция установки организма на карту

Аргументы

pixel↩ _	указатель на организм, который следует установить

См. определение в файле Map.cpp строка 380

5.9.3.26 SetPoison()

void Map::SetPoison (
int numberOfPoison)

Функция распостранения яда по карте

Аргументы

numberOfPoison	количество яда, которое должно находиться на карте

См. определение в файле Map.cpp строка 96

5.9.3.27 Swap()

```
void Map::Swap (
            Hexagon * hexagon1,
            Hexagon * hexagon2 )
```

Функция обмена координатами

Аргументы

hexagon1	указатель на гекс, с которыми нужно поменяться значениями координат
hexagon2	указатель на гекс, с которыми нужно поменяться значениями координат

См. определение в файле Map.cpp строка 386

5.9.3.28 Update()

```
void Map::Update ( )
```

Функция обновления карты

См. определение в файле Map.cpp строка 263

5.9.3.29 UploadFromFile()

```
void Map::UploadFromFile (
            int numberOfEvolution,
            size_t newWidth,
            size_t newHeight )
```

Функция загрузки из файла

Аргументы

numberOfEvolution	номерэволюции, которую следует загрузить
newWidth	ширина карты, которую следует создать
newHeight	высота карты, которую следует создать

См. определение в файле Map.cpp строка 427

Объявления и описания членов классов находятся в файлах:

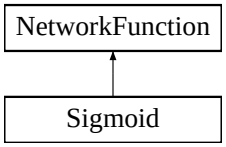
- Map.hpp
- Map.cpp

5.10 Класс NetworkFunction

Родительский класс функции

#include <NetworkFunction.hpp>

Граф наследования:NetworkFunction:



Открытые члены

- NetworkFunction ()=default

  Конструктор по умочанию
- virtual ~NetworkFunction ()=default

  Деструктор по умолчанию
- virtual double Process (const double x)

  Функция получения значения функции
- virtual double Derivative (double x)

  Функция получения значения производной функции

5.10.1 Подробное описание

Родительский класс функции

См. определение в файле NetworkFunction.hpp строка 8

5.10.2 Конструктор(ы)

5.10.2.1 NetworkFunction()

NetworkFunction::NetworkFunction ( ) [default]

Конструктор по умочанию

5.10.2.2 ~NetworkFunction()

virtual NetworkFunction::~NetworkFunction ( ) [virtual], [default]

Деструктор по умолчанию

5.10.3 Методы

5.10.3.1 Derivative()

virtual double NetworkFunction::Derivative (
double x) [inline], [virtual]

Функция получения значения производной функции

Аргументы

x	значение аргумента

Возвращает

значение производной функции

Переопределяется в Sigmoid.

См. определение в файле NetworkFunction.hpp строка 25

5.10.3.2 Process()

virtual double NetworkFunction::Process (
const double x) [inline], [virtual]

Функция получения значения функции

Аргументы

x	значение аргумента

Возвращает

значение функции

Переопределяется в Sigmoid.

См. определение в файле NetworkFunction.hpp строка 18

Объявления и описания членов класса находятся в файле:

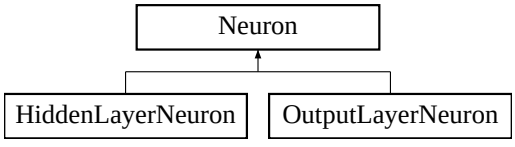
- NetworkFunction.hpp

5.11 Класс Neuron

Класс нейрона

#include <Neuron.hpp>

Граф наследования:Neuron:



Открытые члены

- Neuron ()
  Конструктор по умочанию
- Neuron (NetworkFunction *func)
  Конструктор, создающий нейрон с заданной функцией
- Neuron (std::vector< Neuron *> &vectorOfNeurons, NetworkFunction *func)
  Конструктор по умолчанию
- Neuron (const Neuron &neuron)
  Конструктор копирования
- virtual ~Neuron ()
  Деструктор по умолчанию
- virtual Link * at (const size_t &index) const
  Функция обращения к связи по индексу
- virtual void Input (double deltaWeight)
  Функция изменения веса нейрона
- virtual std::vector< Link * > GetLinksToNeurons () const
  Функция получения всех связей нейрона
- virtual size_t GetNumOfLinks () const
  Функция получения числа связей нейрона
- virtual size_t GetNumOfInputLinks () const
  Функция получения числа входных связей нейрона
- virtual double GetSumOfWeights () const
  Функция получения веса нейрона
- virtual std::vector< Link * > GetInputLinks () const
  Функция получения входных связей нейрона
- virtual void ResetSumOfWeights ()
  Функция обнуления веса нейрона
- virtual double Process (double x) const
  Функция получения значения функции
- virtual void SetLinkToNeuron (Link *link)
  Функция добавления выходной связи
- void SetSumOfWeights (double x)
  Функция установки веса нейрона
- virtual void SetInputLink (Link *link)
  Функция добавления входной связи
- const Json getJson () const
  Функция преобразования данных класса в формат Json.

5.11.1 Подробное описание

Класс нейрона

См. определение в файле Neuron.hpp строка 15

5.11.2 Конструктор(ы)

5.11.2.1 Neuron() [1/4]

```
Neuron::Neuron ( )
```

Конструктор по умочанию

См. определение в файле Neuron.cpp строка 3

5.11.2.2 Neuron() [2/4]

```
Neuron::Neuron (
            NetworkFunction * func )    [explicit]
```

Конструктор, создающий нейрон с заданной функцией

func функция, с которой будет создан нейрон

См. определение в файле Neuron.cpp строка 9

5.11.2.3 Neuron() [3/4]

```
Neuron::Neuron (
            std::vector< Neuron *> & vectorOfNeurons,
            NetworkFunction * func )
```

Конструктор по умолчанию

Аргументы

vectorOfNeurons	вектор нейронов, с которыми соединен нейрон
func	функция нейрона

См. определение в файле Neuron.cpp строка 14

5.11.2.4 Neuron() [4/4]

```
Neuron::Neuron (
            const Neuron & neuron )
```

Конструктор копирования

Аргументы

neuron	нейрон, который следует скопировать

См. определение в файле Neuron.cpp строка 26

5.11.2.5 ~Neuron()

Neuron::~Neuron ( ) [virtual]

Деструктор по умолчанию

См. определение в файле Neuron.cpp строка 34

5.11.3 Методы

5.11.3.1 at()

Link * Neuron::at (
const size_t & index) const [virtual]

Функция обращения к связи по индексу

Аргументы

index	индекс связи, к которой нужно обратиться

Возвращает

связь

См. определение в файле Neuron.cpp строка 39

5.11.3.2 GetInputLinks()

std::vector< Link * > Neuron::GetInputLinks ( ) const [virtual]

Функция получения входных связей нейрона

Возвращает

все входные связи нейрона

См. определение в файле Neuron.cpp строка 69

5.11.3.3 getJson()

const Json Neuron::getJson ( ) const

Функция преобразования данных класса в формат Json.

Возвращает

данные класса в формате Json

См. определение в файле Neuron.cpp строка 100

5.11.3.4 GetLinksToNeurons()

std::vector< Link * > Neuron::GetLinksToNeurons ( ) const [virtual]

Функция получения всех связей нейрона

Возвращает

все связи нейрона

См. определение в файле Neuron.cpp строка 49

5.11.3.5 GetNumOfInputLinks()

size_t Neuron::GetNumOfInputLinks ( ) const [virtual]

Функция получения числа входных связей нейрона

Возвращает

число входных связей нейрона

См. определение в файле Neuron.cpp строка 59

5.11.3.6 GetNumOfLinks()

size_t Neuron::GetNumOfLinks ( ) const [virtual]

Функция получения числа связей нейрона

Возвращает

число связей нейрона

См. определение в файле Neuron.cpp строка 54

5.11.3.7 GetSumOfWeights()

double Neuron::GetSumOfWeights ( ) const [virtual]

Функция получения веса нейрона

вес нейрона

См. определение в файле Neuron.cpp строка 64

5.11.3.8 Input()

void Neuron::Input (
double deltaWeight) [virtual]

Функция изменения веса нейрона

Аргументы

deltaWeight	значение, на котрое следует изменить вес нейрона

См. определение в файле Neuron.cpp строка 44

5.11.3.9 Process()

double Neuron::Process (
double x) const [virtual]

Функция получения значения функции

Аргументы

x	значение аргумента

Возвращает

значение функции

См. определение в файле Neuron.cpp строка 79

5.11.3.10 ResetSumOfWeights()

void Neuron::ResetSumOfWeights ( ) [virtual]

Функция обнуления веса нейрона

См. определение в файле Neuron.cpp строка 74

5.11.3.11 SetInputLink()

void Neuron::SetInputLink (
Link * link) [virtual]

Функция добавления входной связи

Аргументы

link	указатель на связь

См. определение в файле Neuron.cpp строка 95

5.11.3.12 SetLinkToNeuron()

void Neuron::SetLinkToNeuron (
Link * link) [virtual]

Функция добавления выходной связи

Аргументы

link	указатель на связь

См. определение в файле Neuron.cpp строка 85

5.11.3.13 SetSumOfWeights()

void Neuron::SetSumOfWeights (
double x)

Функция установки веса нейрона

Аргументы

x	значение аргумента

См. определение в файле Neuron.cpp строка 90

Объявления и описания членов классов находятся в файлах:

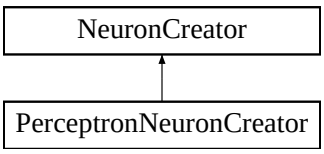
- Neuron.hpp
- Neuron.cpp

5.12 Класс NeuronCreator

Родительский класс нейронной фабрики

#include <NeuronCreator.hpp>

Граф наследования:NeuronCreator:



Открытые члены

- NeuronCreator ()=default

  Конструктор по умочанию
- virtual ~NeuronCreator ()=default

  Деструктор по умолчанию
- virtual Neuron * CreateInputNeuron (std::vector< Neuron *> &inNeuronsLinkTo, NetworkFunction *inNetFunc)=0

  Функция, создающая входной нейрон
- virtual Neuron * CreateOutputNeuron (NetworkFunction *inNetFunc)=0

  Функция, создающая выходной нейрон
- virtual Neuron * CreateHiddenNeuron (std::vector< Neuron *> &inNeuronsLinkTo, NetworkFunction *inNetFunc)=0

  Функция, создающая скрытый нейрон

5.12.1 Подробное описание

Родительский класс нейронной фабрики

См. определение в файле NeuronCreator.hpp строка 9

5.12.2 Конструктор(ы)

5.12.2.1 NeuronCreator()

NeuronCreator::NeuronCreator ( ) [default]

Конструктор по умочанию

5.12.2.2 ~NeuronCreator()

virtual NeuronCreator::~NeuronCreator ( ) [virtual], [default]

Деструктор по умолчанию

5.12.3 Методы

5.12.3.1 CreateHiddenNeuron()

virtual Neuron* NeuronCreator::CreateHiddenNeuron (
std::vector< Neuron *> & inNeuronsLinkTo,
NetworkFunction * inNetFunc) [pure virtual]

Функция, создающая скрытый нейрон

Аргументы

inNeuronsLinkTo	нейроны, с которыми следует соединить новый нейрон
inNetFunc	функция, по которой должен принимать решение нейрон

Замещается в PerceptronNeuronCreator.

5.12.3.2 CreateInputNeuron()

virtual Neuron* NeuronCreator::CreateInputNeuron (
std::vector< Neuron *> & inNeuronsLinkTo,
NetworkFunction * inNetFunc) [pure virtual]

Функция, создающая входной нейрон

Аргументы

inNeuronsLinkTo	нейроны, с которыми следует соединить новый нейрон
inNetFunc	функция, по которой должен принимать решение нейрон

Замещается в PerceptronNeuronCreator.

5.12.3.3 CreateOutputNeuron()

virtual Neuron* NeuronCreator::CreateOutputNeuron (
NetworkFunction * inNetFunc) [pure virtual]

Функция, создающая выходной нейрон

Аргументы

inNetFunc	функция, по которой должен принимать решение нейрон

Замещается в PerceptronNeuronCreator.

Объявления и описания членов класса находятся в файле:

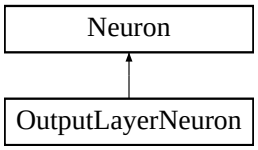
- NeuronCreator.hpp

5.13 Класс OutputLayerNeuron

Выходной нейрон

#include <Neuron.hpp>

Граф наследования:OutputLayerNeuron:



Открытые члены

- OutputLayerNeuron (Neuron *inNeuron)
  Конструктор копирования

5.13.1 Подробное описание

Выходной нейрон

См. определение в файле Neuron.hpp строка 84

5.13.2 Конструктор(ы)

5.13.2.1 OutputLayerNeuron()

```
OutputLayerNeuron::OutputLayerNeuron (
            Neuron * inNeuron )   [inline], [explicit]
```

Конструктор копирования

Аргументы

inNeuron	указатель на нейрон, который следует скопировать

См. определение в файле Neuron.hpp строка 90

Объявления и описания членов класса находятся в файле:

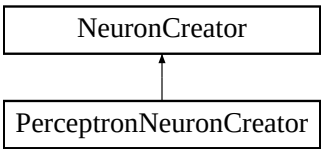
- Neuron.hpp

5.14 Класс PerceptronNeuronCreator

Нейронная фабрика

#include <NeuronCreator.hpp>

Граф наследования:PerceptronNeuronCreator:



Открытые члены

- Neuron * CreateInputNeuron (std::vector< Neuron *> &inNeuronsLinkTo, NetworkFunction *in↩ NetFunc) override

  Функция, создающая входной нейрон
- Neuron * CreateOutputNeuron (NetworkFunction *inNetFunc) override

  Функция, создающая выходной нейрон
- Neuron * CreateHiddenNeuron (std::vector< Neuron *> &inNeuronsLinkTo, NetworkFunction *inNetFunc) override

  Функция, создающая скрытый нейрон

5.14.1 Подробное описание

Нейронная фабрика

См. определение в файле NeuronCreator.hpp строка 30

5.14.2 Методы

5.14.2.1 CreateHiddenNeuron()

```
Neuron* PerceptronNeuronCreator::CreateHiddenNeuron (
            std::vector< Neuron *> & inNeuronsLinkTo,
            NetworkFunction * inNetFunc )   [inline], [override], [virtual]
```

Функция, создающая скрытый нейрон

Аргументы

inNeuronsLinkTo	нейроны, с которыми следует соединить новый нейрон
inNetFunc	функция, по которой должен принимать решение нейрон

Замещает NeuronCreator.

См. определение в файле NeuronCreator.hpp строка 50

5.14.2.2 CreateInputNeuron()

```
Neuron* PerceptronNeuronCreator::CreateInputNeuron (
            std::vector< Neuron *> & inNeuronsLinkTo,
            NetworkFunction * inNetFunc )   [inline], [override], [virtual]
```

Функция, создающая входной нейрон

Аргументы

inNeuronsLinkTo	нейроны, с которыми следует соединить новый нейрон
inNetFunc	функция, по которой должен принимать решение нейрон

Замещает NeuronCreator.

См. определение в файле NeuronCreator.hpp строка 37

5.14.2.3 CreateOutputNeuron()

```
Neuron* PerceptronNeuronCreator::CreateOutputNeuron (
            NetworkFunction * inNetFunc )   [inline], [override], [virtual]
```

Функция, создающая выходной нейрон

Аргументы

inNetFunc	функция, по которой должен принимать решение нейрон

Замещает NeuronCreator.

См. определение в файле NeuronCreator.hpp строка 43

Объявления и описания членов класса находятся в файле:

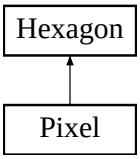
- NeuronCreator.hpp

5.15 Класс Pixel

Класс Организма

#include <Pixel.hpp>

Граф наследования:Pixel:



Открытые члены

- Pixel ()
  Конструктор по умолчанию
- Pixel (double xInPixels, double yInPixels, size_t xInCells, size_t yInCells)
  Конструктор, создающий организм с заданными координатами
- Pixel (double xInPixels, double yInPixels, size_t xInCells, size_t yInCells, const Brain &brainNew)
  Конструктор, создающий организм с заданными координатами и мозгом
- Pixel (double xInPixels, double yInPixels, size_t xInCells, size_t yInCells, double lifesNew, const Brain &brainNew)
  Конструктор, создающий организм с заданными координатами, мозгом и жизнями
- Pixel (double xInPixels, double yInPixels, size_t xInCells, size_t yInCells, double lifesNew, const Brain &brainNew, double medicineNew)
  Конструктор, создающий организм с заданными координатами, мозгом, жизнями и ядовитостью
- Pixel (const Pixel &pixel)
  Конструктор копирования
- std::vector< Hexagon * > LookAround (Map &map) const
  Функция получения стоящих рядом объектов
- void Update (Map &map)
  Функция обновления организма
- void EatingFood (Hexagon *hexToEat, Map &map)
  Функция употребления пищи
- void Move (Map &map, Hexagon *hexToMove)
  Функция передвижения
- void Reproduction (Map &map)
  Функция размножения
- Hexagon * ViewNearbyCells (Map &map, const Type &type)
  Функция просмотра объектов, стоящих вокруг
- unsigned int GetNumberOfLifeIterations () const
  Функция получения числа прожитых итераций
- const Brain & GetBrain () const
  Функция получения мозга организма
- int GetHowMuchFoodAte () const
  Функция получения количества еды, которое съел организм за жизнь
- int GetHowMuchPoisonAte () const
  Функция получения количества яда, которое съел организм за жизнь
- void SetBrain (const Brain &brainNew)

Функция установки мозга

- void ResetNumberOfLifeIterations ()

  Функция обнуления количества прожитых итераций

- void Print (sf::RenderWindow *window) const override

  Функция отрисовки

- const Json getJson () const

  Функция преобразования данных класса в формате Json.

Дополнительные унаследованные члены

5.15.1 Подробное описание

Класс Организма

См. определение в файле Pixel.hpp строка 13

5.15.2 Конструктор(ы)

5.15.2.1 Pixel() [1/6]

Pixel::Pixel ( ) [explicit]

Конструктор по умолчанию

См. определение в файле Pixel.cpp строка 3

5.15.2.2 Pixel() [2/6]

```
Pixel::Pixel (
            double xInPixels,
            double yInPixels,
            size_t xInCells,
            size_t yInCells )
```

Конструктор, создающий организм с заданными координатами

Аргументы

xInPixels	расположение гекса в пикселях по оси Ox на окне
yInPixels	расположение гекса в пикселях по оси Oy на окне
xInCells	номер строки в матрице карты
yInCells	номер столбца в матрице карты

См. определение в файле Pixel.cpp строка 12

5.15.2.3 Pixel() [3/6]

```
Pixel::Pixel (
            double xInPixels,
            double yInPixels,
            size_t xInCells,
            size_t yInCells,
            const Brain & brainNew )
```

Конструктор, создающий организм с заданными координатами и мозгом

Аргументы

xInPixels	расположение гекса в пикселях по оси Ox на окне
yInPixels	расположение гекса в пикселях по оси Oy на окне
xInCells	номер строки в матрице карты
yInCells	номер столбца в матрице карты
brainNew	мозг организма

См. определение в файле Pixel.cpp строка 22

5.15.2.4 Pixel() [4/6]

```
Pixel::Pixel (
            double xInPixels,
            double yInPixels,
            size_t xInCells,
            size_t yInCells,
            double lifesNew,
            const Brain & brainNew )
```

Конструктор, создающий организм с заданными координатами, мозгом и жизнями

Аргументы

xInPixels	расположение гекса в пикселях по оси Ox на окне
yInPixels	расположение гекса в пикселях по оси Oy на окне
xInCells	номер строки в матрице карты
yInCells	номер столбца в матрице карты
lifesNew	количество жизней организма
brainNew	мозг организма

См. определение в файле Pixel.cpp строка 45

5.15.2.5 Pixel() [5/6]

```
Pixel::Pixel (
            double xInPixels,
            double yInPixels,
            size_t xInCells,
            size_t yInCells,
            double lifesNew,
            const Brain & brainNew,
            double medicineNew )
```

Конструктор, создающий организм с заданными координатами, мозгом, жизнями и ядовитостью

Аргументы

xInPixels	расположение гекса в пикселях по оси Ox на окне
yInPixels	расположение гекса в пикселях по оси Oy на окне
xInCells	номер строки в матрице карты
yInCells	номер столбца в матрице карты
lifesNew	количество жизней организма
brainNew	мозг организма
medicineNew	степень ядовитости организма

См. определение в файле Pixel.cpp строка 31

5.15.2.6 Pixel() [6/6]

```
Pixel::Pixel (
            const Pixel & pixel )
```

Конструктор копирования

Аргументы

pixel	организм, которого нужно склонировать

См. определение в файле Pixel.cpp строка 54

5.15.3 Методы

5.15.3.1 EatingFood()

```
void Pixel::EatingFood (
            Hexagon * hexToEat,
            Map & map )
```

Функция употребления пищи

Аргументы

hexToEat	указатель на гекс, который нужно съесть
map	карта, на которой стоит организм

См. определение в файле Pixel.cpp строка 139

5.15.3.2 GetBrain()

const Brain & Pixel::GetBrain ( ) const

Функция получения мозга организма

Возвращает

xInPixels расположение гекса в пикселях по оси Ox на окне

См. определение в файле Pixel.cpp строка 240

5.15.3.3 GetHowMuchFoodAte()

int Pixel::GetHowMuchFoodAte ( ) const

Функция получения количества еды, которое съел организм за жизнь

Возвращает

количество еды, которое съел организм за жизнь

См. определение в файле Pixel.cpp строка 250

5.15.3.4 GetHowMuchPoisonAte()

int Pixel::GetHowMuchPoisonAte ( ) const

Функция получения количества яда, которое съел организм за жизнь

Возвращает

количество яда, которое съел организм за жизнь

См. определение в файле Pixel.cpp строка 254

5.15.3.5 getJson()

const Json Pixel::getJson ( ) const

Функция преобразования данных класса в формате Json.

Возвращает

данные класса в формате Json

См. определение в файле Pixel.cpp строка 274

5.15.3.6 GetNumberOfLifeIterations()

unsigned int Pixel::GetNumberOfLifeIterations ( ) const

Функция получения числа прожитых итераций

Возвращает

число прожитых итераций

См. определение в файле Pixel.cpp строка 235

5.15.3.7 LookAround()

std::vector< Hexagon * > Pixel::LookAround (
Map & map) const

Функция получения стоящих рядом объектов

Аргументы

map	карта, на которой стоит организм

Возвращает

вектор объектов, окружающих организма

См. определение в файле Pixel.cpp строка 62

5.15.3.8 Move()

void Pixel::Move (
Map & map,
Hexagon * hexToMove)

Функция передвижения

Аргументы

hexToMove	указатель на гекс, на который следует переместиться
map	карта, на которой стоит организм

См. определение в файле Pixel.cpp строка 156

5.15.3.9 Print()

void Pixel::Print (
sf::RenderWindow * window) const [override], [virtual]

Функция отрисовки

Аргументы

window	окно, в котором следует отрисовать организм

Переопределяет метод предка Hexagon.

См. определение в файле Pixel.cpp строка 264

5.15.3.10 Reproduction()

void Pixel::Reproduction (
Map & map)

Функция размножения

Аргументы

map	карта, на которой стоит организм

См. определение в файле Pixel.cpp строка 171

5.15.3.11 ResetNumberOfLifeIterations()

void Pixel::ResetNumberOfLifeIterations ( )

Функция обнуления количества прожитых итераций

См. определение в файле Pixel.cpp строка 259

5.15.3.12 SetBrain()

void Pixel::SetBrain (
const Brain & brainNew)

Функция установки мозга

Аргументы

brainNew	мозг, который следует установить

См. определение в файле Pixel.cpp строка 245

5.15.3.13 Update()

void Pixel::Update (
Map & map)

Функция обновления организма

Аргументы

map	карта, на которой стоит организм

См. определение в файле Pixel.cpp строка 126

5.15.3.14 ViewNearbyCells()

Hexagon * Pixel::ViewNearbyCells (
Map & map,
const Type & type)

Функция просмотра объектов, стоящих вокруг

Аргументы

map	карта, на которой стоит организм
type	тип гекса, который требуется найти

См. определение в файле Pixel.cpp строка 193

Объявления и описания членов классов находятся в файлах:

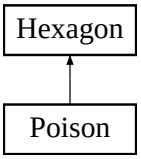
- Pixel.hpp
- Pixel.cpp

5.16 Класс Poison

Класс Яда

#include <Food.hpp>

Граф наследования:Poison:



Открытые члены

- Poison (const double xNew, const double yNew, const size_t CellStrNew, const size_t CellColNew, double Medicine=-(rand() % 12))

  Конструктор, создающий гекс-яд с заданными координатами
- ~Poison () override=default

  Деструктор по умолчанию
- void Print (sf::RenderWindow *window) const override

  Функция отрисовки гекса-яда

Дополнительные унаследованные члены

5.16.1 Подробное описание

Класс Яда

См. определение в файле Food.hpp строка 72

5.16.2 Конструктор(ы)

5.16.2.1 Poison()

```
Poison::Poison (
                const double xNew,
                const double yNew,
                const size_t CellStrNew,
                const size_t CellColNew,
                double Medicine = -(rand() % 12) )    [inline]
```

Конструктор, создающий гекс-яд с заданными координатами

Аргументы

xNew	расположение гекса-яд в пикселях по оси Ox на окне
yNew	расположение гекса-яд в пикселях по оси Oy на окне
CellStrNew	номер строки в матрице карты
CellColNew	номер столбца в матрице карты
Medicine	степнь ядовитости яда

См. определение в файле Food.hpp строка 82

5.16.2.2 ~Poison()

```
Poison::~Poison ( )    [override], [default]
```

Деструктор по умолчанию

5.16.3 Методы

5.16.3.1 Print()

```
void Poison::Print (
                sf::RenderWindow * window ) const    [inline], [override], [virtual]
```

Функция отрисовки гекса-яда

Аргументы

window	окно, в котором нужно отрисовать гекс-яда

Переопределяет метод предка Hexagon.

См. определение в файле Food.hpp строка 91

Объявления и описания членов класса находятся в файле:

- Food.hpp

5.17 Класс Row

#include <Map.hpp>

Открытые члены

- Row ()=default
  Конструктор по умочанию
- ~Row ()=default
  Деструктор по умочанию
- Hexagon *& operator[] (size_t index)
  Оператор обращения по индексу
- const Hexagon * operator[] (size_t index) const
  Оператор обращения по индексу
- void push_back (Hexagon *hex)
  Оператор добавления
- void erase (size_t index)
  Функция удаления
- void insert (Hexagon *hex, size_t index)
  Функция вставки

5.17.1 Подробное описание

См. определение в файле Map.hpp строка 21

5.17.2 Конструктор(ы)

5.17.2.1 Row()

Row::Row ( ) [default]

Конструктор по умочанию

5.17.2.2 ~Row()

Row::~Row ( ) [default]

Деструктор по умочанию

5.17.3 Методы

5.17.3.1 erase()

void Row::erase (
size_t index) [inline]

Функция удаления

Аргументы

index	номер гекса, который нужно убрать

См. определение в файле Map.hpp строка 51

5.17.3.2 insert()

```
void Row::insert (
            Hexagon * hex,
            size_t index )   [inline]
```

Функция вставки

Аргументы

hex	указатель на гекс, который нужно добавить
index	место куда нужно вставить гекс

См. определение в файле Map.hpp строка 58

5.17.3.3 operator[]() [1/2]

```
Hexagon*& Row::operator[] (
            size_t index )   [inline]
```

Оператор обращения по индексу

Аргументы

index	номер элемента в строке

Возвращает

указатель на гекс

См. определение в файле Map.hpp строка 32

5.17.3.4 operator[]() [2/2]

```
const Hexagon* Row::operator[] (
            size_t index ) const   [inline]
```

Оператор обращения по индексу

Аргументы

index	номер элемента в строке

Возвращает

указатель на гекс

См. определение в файле Map.hpp строка 39

5.17.3.5 push_back()

void Row::push_back (
Hexagon * hex) [inline]

Оператор добавления

Аргументы

hex	указатель на гекс, который нужно добавить

См. определение в файле Map.hpp строка 45

Объявления и описания членов класса находятся в файле:

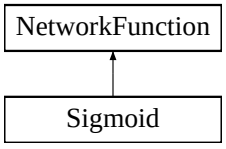
- Map.hpp

5.18 Класс Sigmoid

Класс сигмоидной функции

#include <NetworkFunction.hpp>

Граф наследования:Sigmoid:



Открытые члены

- double Process (const double x) override

  Функция получения значения функции
- double Derivative (const double x) override

  Функция получения значения производной функции

5.18.1 Подробное описание

Класс сигмоидной функции

См. определение в файле NetworkFunction.hpp строка 32

5.18.2 Методы

5.18.2.1 Derivative()

double Sigmoid::Derivative (
const double x) [inline], [override], [virtual]

Функция получения значения производной функции

Аргументы

x	значение аргумента

Возвращает

значение производной функции

Переопределяет метод предка NetworkFunction.

См. определение в файле NetworkFunction.hpp строка 46

5.18.2.2 Process()

double Sigmoid::Process (
const double x) [inline], [override], [virtual]

Функция получения значения функции

Аргументы

x	значение аргумента

Возвращает

значение функции

Переопределяет метод предка NetworkFunction.

См. определение в файле NetworkFunction.hpp строка 39

Объявления и описания членов класса находятся в файле:

- NetworkFunction.hpp

5.19 Класс TrainAlgorithm

Класс Алгоритма тренировки для мозга

#include <TrainAlgorithm.hpp>

Открытые члены

- TrainAlgorithm ()=default

  Конструктор по умочанию
- ~TrainAlgorithm ()=default

  Деструктор по умолчанию
- TrainAlgorithm (Brain *br)

  Конструктор, создающий алгоритм тренировки для мозга
- void Train (Brain *br)

  Функция, осуществляющая тренировку мозга
- void WeightsInitialization ()

  Функция, осуществляющая тренировку мозга
- void WeightsInitialization (const Json &)

  Функция, осуществляющая заполнение весов в соответствии со значениями в Json.

5.19.1 Подробное описание

Класс Алгоритма тренировки для мозга

См. определение в файле TrainAlgorithm.hpp строка 13

5.19.2 Конструктор(ы)

5.19.2.1 TrainAlgorithm() [1/2]

TrainAlgorithm::TrainAlgorithm ( ) [default]

Конструктор по умочанию

5.19.2.2 ~TrainAlgorithm()

TrainAlgorithm::~TrainAlgorithm ( ) [default]

Деструктор по умолчанию

5.19.2.3 TrainAlgorithm() [2/2]

TrainAlgorithm::TrainAlgorithm (
Brain * br) [explicit]

Конструктор, создающий алгоритм тренировки для мозга

Аргументы

br	указатель на мозг

См. определение в файле TrainAlgorithm.cpp строка 53

5.19.3 Методы

5.19.3.1 Train()

```
void TrainAlgorithm::Train (
            Brain * br )
```

Функция, осуществляющая тренировку мозга

Аргументы

br	указатель на мозг

См. определение в файле TrainAlgorithm.cpp строка 58

5.19.3.2 WeightsInitialization() [1/2]

```
void TrainAlgorithm::WeightsInitialization ( )
```

Функция, осуществляющая тренировку мозга

См. определение в файле TrainAlgorithm.cpp строка 30

5.19.3.3 WeightsInitialization() [2/2]

```
void TrainAlgorithm::WeightsInitialization (
            const Json & object )
```

Функция, осуществляющая заполнение весов в соответствии со значениями в Json.

значения весов

См. определение в файле TrainAlgorithm.cpp строка 36

Объявления и описания членов классов находятся в файлах:

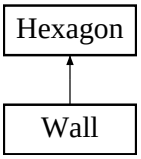
- TrainAlgorithm.hpp
- TrainAlgorithm.cpp

5.20 Класс Wall

Класс Стены

#include <Food.hpp>

Граф наследования:Wall:



Открытые члены

- Wall ()

  Конструктор, создающий гекс-стену
- ~Wall () override=default

  Деструктор по умолчанию

Дополнительные унаследованные члены

5.20.1 Подробное описание

Класс Стены

См. определение в файле Food.hpp строка 103

5.20.2 Конструктор(ы)

5.20.2.1 Wall()

Wall::Wall ( ) [inline]

Конструктор, создающий гекс-стену

См. определение в файле Food.hpp строка 108

5.20.2.2 ~Wall()

Wall::~Wall ( ) [override], [default]

Деструктор по умолчанию

Объявления и описания членов класса находятся в файле:

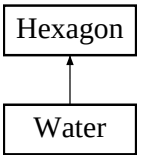
- Food.hpp

5.21 Класс Water

Класс Воды

#include <Food.hpp>

Граф наследования:Water:



Открытые члены

- Water (const double xNew, const double yNew, const size_t CellStrNew, const size_t CellColNew)
  Конструктор, создающий гекс-воды с заданными координатами
- ~Water () override=default
  Деструктор по умолчанию
- void Print (sf::RenderWindow *window) const override
  Функция отрисовки гекса-воды

Дополнительные унаследованные члены

5.21.1 Подробное описание

Класс Воды

См. определение в файле Food.hpp строка 42

5.21.2 Конструктор(ы)

5.21.2.1 Water()

```
Water::Water (
            const double xNew,
            const double yNew,
            const size_t CellStrNew,
            const size_t CellColNew ) [inline]
```

Конструктор, создающий гекс-воды с заданными координатами

Аргументы

xNew	расположение гекса-воды в пикселях по оси Ox на окне
yNew	расположение гекса-воды в пикселях по оси Oy на окне
CellStrNew	номер строки в матрице карты
CellColNew	номер столбца в матрице карты

См. определение в файле Food.hpp строка 51

5.21.2.2 ~Water()

Water::~Water ( ) [override], [default]

Деструктор по умолчанию

5.21.3 Методы

5.21.3.1 Print()

void Water::Print (
sf::RenderWindow * window) const [inline], [override], [virtual]

Функция отрисовки гекса-воды

Аргументы

window	окно, в котором нужно отрисовать гекс-воду

Переопределяет метод предка Hexagon.

См. определение в файле Food.hpp строка 60

Объявления и описания членов класса находятся в файле:

- Food.hpp

Глава 6

Файлы

6.1 Файл Brain.cpp

```
#include "Brain.hpp"
#include "Hexagon.hpp"
```

6.2 Файл Brain.hpp

```
#include <algorithm>
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "NeuronCreator.hpp"
#include "TrainAlgorithm.hpp"
#include <nlohmann/json.hpp>
```

Классы

- class Brain

  Класс, описывающий мозг организма

Определения типов

- using Json = nlohmann::json

6.2.1 Типы

6.2.1.1 Json

using Json = nlohmann::json

См. определение в файле Brain.hpp строка 15

6.3 Файл Button.cpp

#include "Button.hpp"

6.4 Файл Button.hpp

#include <cmath>
#include <SFML/Graphics/CircleShape.hpp>
#include <SFML/Graphics/RenderWindow.hpp>

Классы

- class Button

  Класс кнопки

6.5 Файл Evolution.cpp

#include "Evolution.hpp"

6.6 Файл Evolution.hpp

#include <thread>
#include <SFML/Window.hpp>
#include "Keyboard.hpp"
#include "Map.hpp"
#include "Pixel.hpp"
#include "Button.hpp"

Классы

- class Evolution

  Класс Эволюции

6.7 Файл Food.hpp

```
#include "Hexagon.hpp"
#include <nlohmann/json.hpp>
```

Классы

- class Food
  Класс Еды
- class Water
  Класс Воды
- class Poison
  Класс Яда
- class Wall
  Класс Стены

Определения типов

- using Json = nlohmann::json

6.7.1 Типы

6.7.1.1 Json

using Json = nlohmann::json

См. определение в файле Food.hpp строка 8

6.8 Файл Hexagon.cpp

```
#include "Hexagon.hpp"
#include "Map.hpp"
```

6.9 Файл Hexagon.hpp

```
#include <random>
#include <SFML/Graphics/CircleShape.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include "Brain.hpp"
```

Классы

- class Hexagon

  Родительский класс, описывающий шестиугольник(гекс)

6.10 Файл Keyboard.cpp

```
#include "Keyboard.hpp"
```

6.11 Файл Keyboard.hpp

```
#include <map>
#include <SFML/Graphics.hpp>
```

Классы

- class Keyboard

  Класс Клавиатуры

6.12 Файл Link.cpp

```
#include "Link.hpp"
```

6.13 Файл Link.hpp

Классы

- class Link

  Класс Связи между нейронами

6.14 Файл Map.cpp

```
#include "Map.hpp"
#include <nlohmann/json.hpp>
#include "Pixel.hpp"
```

Определения типов

- using Json = nlohmann::json

Функции

- bool Difference (Pixel *a, Pixel *b)

6.14.1 Типы

6.14.1.1 Json

```
using Json = nlohmann::json
```

См. определение в файле Map.cpp строка 5

6.14.2 Функции

6.14.2.1 Difference()

```
bool Difference (
                Pixel * a,
                Pixel * b )
```

См. определение в файле Map.cpp строка 116

6.15 Файл Map.hpp

```
#include <iomanip>
#include <memory>
#include <string>
#include <thread>
#include <vector>
#include <boost/filesystem.hpp>
#include <SFML/Graphics.hpp>
#include <nlohmann/json.hpp>
#include "Food.hpp"
#include "Hexagon.hpp"
```

Классы

- class Row
- class Map

Определения типов

- using Json = nlohmann::json

6.15.1 Типы

6.15.1.1 Json

using Json = nlohmann::json

См. определение в файле Map.hpp строка 17

6.16 Файл NetworkFunction.hpp

#include <cmath>

Классы

- class NetworkFunction
  Родительский класс функции
- class Sigmoid
  Класс сигмоидной функции

6.17 Файл Neuron.cpp

#include "Neuron.hpp"

6.18 Файл Neuron.hpp

#include <vector>
#include <nlohmann/json.hpp>
#include "Link.hpp"
#include "NetworkFunction.hpp"

Классы

- class Neuron
  Класс нейрона
- class OutputLayerNeuron
  Выходной нейрон
- class HiddenLayerNeuron
  Скрытый нейрон

Определения типов

- using Json = nlohmann::json

6.18.1 Типы

6.18.1.1 Json

using Json = nlohmann::json

См. определение в файле Neuron.hpp строка 11

6.19 Файл NeuronCreator.hpp

#include "Neuron.hpp"
#include "NetworkFunction.hpp"

Классы

- class NeuronCreator

  Родительский класс нейронной фабрики
- class PerceptronNeuronCreator

  Нейронная фабрика

6.20 Файл Pixel.cpp

#include "Pixel.hpp"

6.21 Файл Pixel.hpp

#include <algorithm>
#include "Hexagon.hpp"
#include "Food.hpp"
#include "Map.hpp"
#include <nlohmann/json.hpp>

Классы

- class Pixel

  Класс Организма

Определения типов

- using Json = nlohmann::json

6.21.1 Типы

6.21.1.1 Json

using Json = nlohmann::json

См. определение в файле Pixel.hpp строка 10

6.22 Файл TrainAlgorithm.cpp

```
#include "Brain.hpp"
#include "TrainAlgorithm.hpp"
```

Функции

- double doublerand (double a, double b)

6.22.1 Функции

6.22.1.1 doublerand()

```
double doublerand (
                double a,
                double b )
```

См. определение в файле TrainAlgorithm.cpp строка 5

6.23 Файл TrainAlgorithm.hpp

```
#include <vector>
#include <nlohmann/json.hpp>
```

Классы

- class TrainAlgorithm

  Класс Алгоритма тренировки для мозга

Определения типов

- using Json = nlohmann::json

6.23.1 Типы

6.23.1.1 Json

using Json = nlohmann::json

См. определение в файле TrainAlgorithm.hpp строка 7

Предметный указатель