Representation of Polynomials (Documentation)

Compiling program

```
$: g++ main.cpp -o main polynomial.cpp functions.cpp
```

Running program

for Linux

```
$: ./main
```

for Windows

```
> main.exe
```

Class Polynomial

Class Polynomial is a class that represents a polynomial using the array as a set of arguments of each element of polynomial and has such methods and constructors:

```
// Basic constructor
Polynomial();

// Constructor with arguments
Polynomial(int size, double arr[] = {}) : size(size), array(arr){};

// Constructor with arguments
Polynomial(double arg, int size);
~Polynomial(){};


// Operations to perform arithmetic operations with polynomials.
Polynomial operator+(const Polynomial &other);
Polynomial operator-(const Polynomial &other);
Polynomial operator*(const Polynomial &other);
Polynomial operator*(const int &number);
Polynomial operator=(const Polynomial &other);
Polynomial &operator+=(const Polynomial &right);
Polynomial &operator-=(const Polynomial &right);


// Operators to perform logic operations with polynomials
bool operator==(const Polynomial &other);
bool operator!=(const Polynomial &other);
```

```
// Method to exponent the polynomial by a number;
Polynomial pow(int n);
// Method to defferentiate th polynomial.
Polynomial diff();
// Method to integrate polynomial.
Polynomial integrate();
// Method to combine two polynomials into one.s
Polynomial combine(const Polynomial &other);

// Returns size of polynomial.
int getSize();
// Method to make a negative polynomial.
void negation();
// Method to check if polynomial equals to 0.
bool is_zero();
// Method to evaluate polynomial using horner algorithm.
double eval_by_Horner(const double &x);

// Method to return an array with arguments of the each element of polynomial.
string toString();
// Method to return a representation of polynomial.
string representation();
```

Constructors

```
// Basic constructor
Polynomial();
```

> Basic constructor, creates an empty polynomial that has a nullptr as a pointer to array.

```
// Constructor with arguments.
Polynomial(int size, double arr[] = {}) : size(size), array(arr){};
```

> Creates a polynomial with given size, and its arguments of elements are taken from the given array.

Parameters:

- size - size of the polynomial.
- arr - array that will be passed as arguments of polynomial elements

---

...

...

```
Polynomial(double arg, int size);
```

Creates polynomial with all element assigned to 0, except the last one which will be assigned as an argument arg.

Operators

```
// Returns new instance of class which is a product of addition
Polynomial operator+(const Polynomial &other);
// Returns new instance of class which is a product of substraction
Polynomial operator-(const Polynomial &other);
// Returns new instance of class which is a product of
// multiplication by another polynomial
Polynomial operator*(const Polynomial &other);
// Returns new instance of class which is a product of
// multiplication by a number
Polynomial operator*(const int &number);
// Returns new instance of class which is a product of equating
Polynomial operator=(const Polynomial &other);
// Returns new instance of class which is a product of addind and equating
Polynomial &operator+=(const Polynomial &right);
// Returns new instance of class which is a product of substracting and equating
Polynomial &operator-=(const Polynomial &right);
// Returns true is two polynomials are equal.
bool operator==(const Polynomial &other);
// Returns true if two polynimials are not equal.
bool operator!=(const Polynomial &other);
```

Public methods

```
Polynomial pow(int n);
```

Exponenting each element of the polynomial by a number.

Parameters:

- n - number of exponentiation grade.

```
Polynomial diff();
```

Differentiating each element of the polynomial.

```
Polynomial integrate();
```

Integrating each element of the polynomial.

```
Polynomial combine(const Polynomial &other);
```

Combining two polynomials. Each variable from the first polynomial will be replaced by second polynomial.

Parameters:

- other - second polynomial that will replace all variables of the first polynomial

---

```
int getSize();
```

Returns size of polynomial.

```
void negation();
```

Makes a polynomial negative.

```
bool is_zero();
```

Checks if polynomial equals to 0.

```
double eval_by_Horner(const double &x);
```

Evaluating polynomial using horner algorithm.

Parameters:

- x - value that will replace all variable occurances in polynomial.

```
string toString();
```

Returns an array with arguments of the each element of polynomial.

```
string representation();
```

Returns a representation of polynomial.
