**Algorytmy i struktury danych  
Projekt zaliczeniowy**

**Temat: „Implementacja wielomianów na bazie tablic”**

Autor: Tomasz Różycki

*Wstęp:*

Niniejszy projekt ma na celu zaimplementowanie wielomianów na bazie tablic oraz działań: dodawania, odejmowania, mnożenia i obliczania wartości wielomianu schematem Hornera.

*Definicja wielomianu:*

Wielomian (inaczej suma algebraiczna) – jest to wyrażenie algebraiczne będące sumą jednomianów. Dla nieujemnej liczby całkowitej n *wielomianem stopnia n zmiennej x* jest wyrażenie postaci:

gdzie są współczynnikami wielomianu oraz

*Algorytm Hornera:*

Jest to sposób obliczania wartości wielomianu dla danej wartości argumentu **wykorzystujący minimalną liczbę mnożeń**. Mając wielomian w postaci:

To obliczając jego wartość dla zadanego ***x*** bezpośrednio z zadanego wzoru należy wykonać mnożeń i dodawań.

Tymczasem proste przekształcenie:

Sprawia, że wystarczy jedynie ***n*** mnożeń i ***n*** dodawań.

*Dodawanie, odejmowanie i mnożenie wielomianów:*

Dodawanie i odejmowanie są to metody intuicyjne, wykonujemy działania na współczynnikach przy odpowiednich potęgach zmiennej *x.* W przypadku mnożenia warto zauważyć, że stopień nowopowstałego wielomianu jest sumą stopni mnożonych wielomianów. Operacje te można wykonywać w sposób analogiczny do operacji na pozycyjnym systemie liczbowym.

*Opis programu:*

Definicja typów:

class Polynomial

{

private:

    double \*coefficient\_arr;

    int arr\_size; //rozmiar tablicy jest o jeden wiekszy od stopnia wielomianu

    //przyklad:

    //int stopien = 2

    //x^2 + x + c --> trzy wspolczynniki

public:

    //konstruktor - tworzy wielomian zainicjalizowany na zero

    Polynomial(int degree);

    //konstruktor - tworzy wielomian i inicjalizuje wartosci

    Polynomial(double \*arr, int coefficient);

    //destruktor

    ~Polynomial();

    //dodaje jeden wielomian do drugiego

    Polynomial Add(Polynomial &other);

    //odejmuje wielomiany

    Polynomial Substract(Polynomial &other);

    //mnozenie wielomanow

    Polynomial Multiply(Polynomial &other);

    //zwraca stopien wielomanu

    int degree();

    //obliczanie wartosci wielomianu metoda Hornera

    double Horner(double value);

    //wyswietla wielomian

    void displayPolynomial();

};

Najważniejsze algorytmy:

1. Dodawanie i odejmowanie

Polynomial Polynomial::Add(Polynomial &other)

{

    //sprawdzamy ktory wielomian ma wiekszy stopien

    int resultSize = (arr\_size >= other.arr\_size) ? arr\_size : other.arr\_size;

    //tworzymy nowy obiekt, stopien o 1 mniejszy od arr\_size

    Polynomial result(resultSize - 1);

    if (arr\_size >= other.arr\_size)

    {

        for (int i = 0; i < other.arr\_size; i++)

        {

            //jesli other.arr ma mniejszy lub rowny stopien to dodaje tylko wspolczynniki do rozmiaru other.arrsize

            result.coefficient\_arr[i] = coefficient\_arr[i] + other.coefficient\_arr[i];

        }

        for (int i = other.arr\_size; i < arr\_size; i++)

        {

            //pozosotale wspolczynniki przepisuje

            result.coefficient\_arr[i] = coefficient\_arr[i];

        }

    }

    else

    {

        for (int i = 0; i < arr\_size; i++)

        {

            result.coefficient\_arr[i] = coefficient\_arr[i] + other.coefficient\_arr[i];

        }

        for (int i = arr\_size; i < other.arr\_size; i++)

        {

            result.coefficient\_arr[i] = other.coefficient\_arr[i];

        }

    }

    return result;

}

Odejmowanie zostało zaimplementowane w sposób analogiczny do dodawania, lecz zamiast dodawać odpowiednie współczynniki, odejmujemy je. Wymusiło to zmianę znaku + na – w trzech miejscach.

**Złożoność:**

Niech N = stopień większego wielomianu + 1

O(N)

1. *Mnożenie*

Polynomial Polynomial::Multiply(Polynomial &other)

{

    //najwyzsza potega wielomianu to suma rozmiaru dwoch tablic - 2(bo wspolczynnik x^0)

    int result\_degree = arr\_size + other.arr\_size - 2;

    //tworzymy nowy obiekt

    Polynomial result(result\_degree);

    //mnozenie polega na wymnozeniu kazdego elementu wielomianu\_1 z wielomianem\_2

    for (int i = 0; i < arr\_size; i++) //przegladamy wszystkie elementy wielomianu\_1

    {

        for (int j = 0; j < other.arr\_size; j++) //przegladamy wszyskie elementy wielomianu\_2

        {

            //[i+j] poniewaz przy mnozeniu poteg, jest to suma wspolczynnikow

            result.coefficient\_arr[i + j] += (coefficient\_arr[i] \* other.coefficient\_arr[j]);

        }

    }

    return result;

}

**Złożoność:**

Oznaczmy rozmiar tablicy przechowujący współczynniki wielomianu:

1. n = arr\_size
2. m = other.arr\_size

O(n \*m)

1. *Algorytm Hornera*
2. double Polynomial::Horner(double x)
3. {
4. double result = 0;
5. if (arr\_size == 1){
6. result = coefficient\_arr[0];
7. }
8. else
9. {
10. result = coefficient\_arr[arr\_size - 1];
11. for (int i = arr\_size - 2; i >= 0; i--)
12. {
13. //a\_n \* x + a\_(n-1)
14. result = result \* x + coefficient\_arr[i];
15. }
16. }
17. return result;
18. }

**Złożoność:**

Niech N = rozmiar tablicy współczynników

O(N)

**Analiza algorytmu:**

Przypomnijmy uproszczenie postaci wielomianu:

Wykonywanie działań zaczynamy od „najbardziej zagłębionego” nawiasu, mnożymy współczynnik przy najwyższej potędze razy argument x oraz dodajemy współczynnik przy o jeden mniejszej potędze.

*Kompilacja i uruchomienie:*

g++ main.cpp library.cpp -o main

main.exe

Następnie program w sposób intuicyjny prosi użytkownika o wprowadzenie danych.