**Dokumentacja c.d.**

**Struktury danych, procedury i funkcje:**

**Tablice:**

* **double** x[n**+**1]; - tablica 1-dno wymiarowa przechowująca węzły od x0 do xn
* **double** l[n**+**1][n**+**1]; - tablica 2 wymiarowa przechowująca współczynniki mnożników Lagrange’a
* **double** c[n**+**1]; - tablica 1-dno wymiarowa przechowująca wartości funkcji interpolowanej
* **double** w[n**+**1]; - tablica 1-dno wymiarowa przechowująca wielomian w postaci Lagrange’a

**Procedury:**

Jedna ważna pętla wykonująca obliczenie mnożników Lagrange’a:

**for** (i**=**0; i**<=**n; i**++**){

**for** (k**=**0; k**<=**n; k**++**){

**if** (i**!=**k){

**for** (p**=**n; p**>**0; p**--**){

        l[i][p] **=** l[i][p**-**1];

      }

      l[i][0] **=** 0;

**for** (p**=**0; p**<=**n**-**1; p**++**){

        l[i][p] **-=** x[k] **\*** l[i][p**+**1];

      }

**for** (p**=**0; p**<=**n; p**++**){

        l[i][p] **/=** x[i] **-** x[k];

      }

    }

  }

}

Pętla na wstępie otrzymuje tablicę dwuwymiarową l, wypełnioną na miejscach l[i][0] wartościami 1 a reszta wartości równa 0 dla i z przedziału od 0 do n.

Pętla zewnętrzna wypełnia tablicę l po indeksach dolnych według wzoru podanego w zadaniu gdzie pętla z zmienną k spełnia funkcję iloczynu mnożącego w 3 kolejnych pętlach wartości wielomianów.

W klauzuli warunkowej if algorytm sprawdza czy i jest różne od k po czym w przypadku TRUE wykonuje 3 wewnętrzne pętle które to:

1. Przesuwa wartości tablicy l w danym indeksie i w prawo o 1 miejsce symulując mnożenie wielomianów (zwiększanie stopni wielomianu) po czym pierwszy element jest zerowany
2. Pętla oblicza liczniki ułamka i podpisuje je w odpowiednim stopniu tablicy l poprzez mnożenie elementu na prawo od niej (większego stopniem) z x o indeksie k.
3. Dzieli każdy licznik wpisany w tablicę l.

**Wejście/wyjście:**

**Wejście:**

* **n** – liczba naturalna określająca maksymalny stopień wielomianu które ma być mniejsze od NMAX zabezpieczone w pętli typu do while z warunkiem n>NMAX zdefiniowanym w programie
* **a,b** – zmienne służące do obliczania węzłów, zabezpieczone przez pętle typu do while z warunkiem a>=b
* **c[n+1]** – wartości funkcji interpolowanej A0 do An, programistycznie zabezpieczone przez wczytanie scanf z formatem %lf
* pętla typu do while wczytująca znaki i w przypadku podania t przenosząca do etykiety przez podaniem wartości funkcji interpolowanej

**Wyjście:**

* Wielomian w postaci Lagrange’a

**Złożoność programu:**

Złożoność głównej pętli wynosi Θ (n3) a obliczenie wielomianu wynikowego przy wczytaniu zestawu wartości A wynosi Θ (n2) co daje wynikową złożoność w wymiarze Θ (n3).

**Kod:**

**#include <stdio.h>**

**#define NMAX 1000**

**int** main(**void**){

**int** n,i,j,k,p;

**double** a,b;

**do** {

  printf("Podaj n: ");

  scanf("%i", **&**n);

} **while** (n**>**NMAX);

**do** {

  printf("Podaj a i b z zalozeniem a<b\na: ");

  scanf("%lf", **&**a);

  printf("b: ");

  scanf("%lf", **&**b);

} **while** (a**>=**b);

**double** h **=** (b**-**a)**/**n;

**double** x[n**+**1];

**for** (i**=**0; i**<=**n; i**++**){

  x[i] **=** a **+** i **\*** h;

*// printf("x[%i] = %lf\n", i, x[i]);*

}

**double** l[n**+**1][n**+**1];

**for** (i**=**0; i**<=**n; i**++**){

**for** (j**=**1; j**<=**n; j**++**){

    l[i][j] **=** 0;

  }

  l[i][0] **=** 1;

}

**for** (i**=**0; i**<=**n; i**++**){

**for** (k**=**0; k**<=**n; k**++**){

**if** (i**!=**k){

**for** (p**=**n; p**>**0; p**--**){

        l[i][p] **=** l[i][p**-**1];

      }

      l[i][0] **=** 0;

**for** (p**=**0; p**<=**n**-**1; p**++**){

        l[i][p] **-=** x[k] **\*** l[i][p**+**1];

      }

**for** (p**=**0; p**<=**n; p**++**){

        l[i][p] **/=** x[i] **-** x[k];

      }

    }

  }

}

*/\* wyswietlanie l[][]*

*for (i=0; i<=n; i++){*

*printf("l[%i]: ", i);*

*for (j=n; j>=0; j--){*

*printf("%.3lfx^%i ", l[i][j],j);*

*}*

*printf("\n");*

*}\*/*

**double** c[n**+**1], w[n**+**1];

**char** o;

wczytaj:

**for** (i**=**0; i**<=**n; i**++**){

  printf("Podaj A[%i]: ", i);

  scanf("%lf", **&**c[i]);

}

**for** (i**=**0; i**<=**n; i**++**){

  w[i] **=** 0;

}

**for** (i**=**0; i**<=**n; i**++**){

**for** (j**=**0; j**<=**n; j**++**){

    w[j] **+=** ( c[i] **\*** l[i][j] );

  }

}

**for** (i**=**n; i**>=**0; i**--**){

  printf("%.3lfx^%i ", w[i], i);

}

printf("\nWprowadzac dane ponownie? (t/n) ");

**do** {

  scanf("%c", **&**o);

} **while** (o**!=**'t' **&&** o**!=**'n');

**if** (o**==**'t'){

**goto** wczytaj;

}

putchar('\n');

**return** 0;

}

**Wydruki:**

﻿Test zabezpieczenia dla n:

Podaj n: 50000

Podaj n: 32001

Podaj n: 29999

Podaj a i b z zalozeniem a<b

a:

Test zabezpieczenia dla a i b:

Podaj a i b z zalozeniem a<b

a: 8

b: 1

Podaj a i b z zalozeniem a<b

a: 5

b: 5

Podaj a i b z zalozeniem a<b

a: 8

b: 10

Podaj A[0]:

Test zabezpieczenia dla pytania o powtórne wpisywanie

Wprowadzac dane ponownie? (t/n) p

x

z

c

n

Poprawnie wypełnione dane:

Podaj n: 2

Podaj a i b z zalozeniem a<b

a: 1

b: 3

Podaj A[0]: 5

Podaj A[1]: 1

Podaj A[2]: 2

2.500x^2 -11.500x^1 14.000x^0

Wprowadzac dane ponownie? (t/n) t

Podaj A[0]: 1

Podaj A[1]: 1

Podaj A[2]: 2

0.500x^2 -1.500x^1 2.000x^0

Wprowadzac dane ponownie? (t/n) n