Zadanie 1

Damian Soliński
Piotr Kulas
Marek Skiba

15.10.2012

1 Treść zadania

Ustalić liczbe naturalna n_{max}. Wczytać n $\in$ {0,1,...,n_{max}} oraz wartości A_0,A_1,...,A_n. Nastepnie, dopóki "użytkownik sie nie znudzi", wczytywać x_0 $\in$ **R** i wyznaczać w postaci ogólnej wielomian W = W(x) stopnia co najwyżej n taki, że $W^{(i)}(x_0)=A_i$, i=0,1,...,n.

2 Schemat rozwiazywania zadania

Ustalamy n_{max}. Wczytujemy wartości A_0,A_1,...,A_n, nastepnie wczytujemy x_0. Układ n równań liniowych o n niewiadomych ma postać :
$W^{(0)}(x) = B_n x^n + B_{n-1} x^{n-1} + ... + B_1 x^1 + B_0 x^0 = A_0$
$W^{(1)}(x) = (B_n x^n)^{(1)} + (B_{n-1} x^{n-1})^{(1)} + ... + (B_1 x^1)^{(1)} = A_1$
$W^{(2)}(x) = (B_n x^n)^{(2)} + (B_{n-1} x^{n-1})^{(2)} + ... + (B_2 x^2)^{(2)} = A_2$
.
.
.
$W^{(n-1)}(x) = (B_n x^n)^{(n-1)} + (B_{n-1} x^{n-1})^{(n-1)} = A_{n-1}$
$W^{(n)}(x) = (B_n x^n)^{(n)} = A_n$

Dla każdego działania $(B_k x^k)^{(i)}$ gdzie k<i wynik jest równy 0 i jest pominiete w układzie równań.
Liczac od ostatniego równania możemy łatwo uzyskać wyniki naszych niewiadomych A_0,A_1,...,A_n.

3 Przykładowe rozwiazanie

Niech $n_{max} = 3$. Wczytujemy nastepne dane :

$$A_0=1, A_1=2, A_2=3 A_3=6, x_0=2$$

Powstaje układ równań :

$B_3x^3+B_2x^2+B_1x^1+ B_0x^0 = 1$

$3B_3x^2+2B_2x^1+B_1x^0+ 0 = 2$

$6B_3x^1+2B_2x^0+0+ 0 = 4$

$6B_3x^0+0+0+ 0 = 6$

Podstawiamy x_0 w miejsce x :

$B_3*8+B_2*4+B_1*2+ B_0 = 1$

$3B_3*4+2B_2*2+B_1 = 2$

$6B_3*2+2B_2 = 4$

$6B_3= 6$

Liczymy od końca :

$B_3 = \frac{6}{6} = 1$

$12 * 1 + 2B_2 = 4$

$2B_2 = -8$

$B_2 = -4$

$3*1*4+2*(-4)*2+B_1 = 2$

$12-16+B_1 = 2$

$B_1=6$

$1*8+(-4)*4+6*2+ B_0 = 1$

$8-16+12+B_0=1$

$4+B_0=1$

$B_0=-3$

Rozwiazaniem zadania jest :

$$W(x)=x^3-4x^2+6x-3$$

4 Źródło programu:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define N_MAX 100

double MyPow(double x0, int exp) {
    return exp > 0 ? MyPow(x0, exp - 1) * x0 : 1;
}

void fillMatrix(int n, double **t, double x0) {
    int i, j;
    for (j = 0; j < n; j++) {
        t[0][j] = 1;
        for (i = 1; i < n; i++) {
            if (i + j < n)t[i][j] = t[i - 1][j] * (n - i - j);
        }
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i + j < n)t[i][j] *= MyPow(x0, n - j - i - 1);
        }
    }
}

void resolveMatrix(int n, double **t) {
    int i, j;
    for (i = n - 1; i >= 0; i--) {
        t[i][n] /= t[i][n - i - 1];
        for (j = 0; j < i; j++) {
            t[j][n] -= t[j][n - i - 1] * t[i][n] ;
        }
    }
}

void printMatrix(int n, double **t) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j <= n; j++) {
            printf("%lf ", t[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
```

```
void clearMatrix(int n, double **t, double *A) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            t[i][j] = 0;
        }
        t[i][n] = A[i];
    }
}

void printPolynomial(int n, double **t) {
    int i = n;
    printf("Wynik: W(x) = ");
    while (--i) {
        printf("%lf", t[i][n]);
        printf("x^%d + ", i);
    }
    printf("%lfx^0\n", t[0][n]);
}

int main() {
    double *A, **t, x0;
    int i, n;
    char chce_podac_x[2];

    do {
        printf("Podaj liczbe naturalna z przedziału od 0 do %d: ", N_MAX);
        scanf("%d", &n);
    } while(n < 0 || n > N_MAX);

    n += 1;
    A = malloc(n * sizeof (double));
    for (i = 0; i < n; i++) {
        printf("Podaj A[%d]: ", i);
        scanf("%lf", &A[i]);
    }
    t = malloc(n * sizeof (double*));
    for (i = 0; i < n; i++) {
        t[i] = malloc((n + 1) * sizeof (double));
    }

    do {
        printf("Podaj X0: ");
        scanf("%lf", &x0);
        clearMatrix(n, t, A);
        fillMatrix(n, t, x0);
        resolveMatrix(n, t);
        printPolynomial(n, t);
```

```
        do {
            printf("Chcesz podać kolejne X0? (tak/nie): ");
            scanf("%s", chce_podac_x);
            if(strcmp(chce_podac_x, "tak") == 0 || strcmp(chce_podac_x, "nie") == 0)
        } while(1);

    } while (strcmp(chce_podac_x, "tak") == 0);

    return 0;
}
```

5 Wynik programu:

```
[mskiba@sigma] ~/MetodyObliczeniowe $ gcc zad1.c -o zad1
[mskiba@sigma] ~/MetodyObliczeniowe $ ./zad1
Podaj liczbe naturalna z przedziału od 0 do 100: 101
Podaj liczbe naturalna z przedziału od 0 do 100: -1
Podaj liczbe naturalna z przedziału od 0 do 100: 3
Podaj A[0]: 2
Podaj A[1]: 3
Podaj A[2]: 4
Podaj A[3]: 6
Podaj X0: 2
Wynik: W(x) = 1.000000x^3 + -4.000000x^2 + 7.000000x^1 + -4.000000x^0
Chcesz podać kolejne X0? (tak/nie): s
Chcesz podać kolejne X0? (tak/nie): tak
Podaj X0: 4
Wynik: W(x) = 1.000000x^3 + -10.000000x^2 + 35.000000x^1 + -42.000000x^0
Chcesz podać kolejne X0? (tak/nie): nie
[mskiba@sigma] ~/MetodyObliczeniowe $
```