

DS LAB 3

12/08/2024

Name: Utkarsh Yadav

Roll: 23053172

Sec: CSE37

ADDITION OF POLYNOMIALS

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5
6 void printPolynomial(int** poly, int terms){
7     printf("f(x) = ");
8     for(int i=0; i<terms; i++){
9         if(i>0 && poly[0][i]>0){
10             printf(" + ");
11         }
12
13         if(poly[0][i]>0){
14             if(poly[1][i]==0){
15                 printf("%d", poly[0][i]);
16             } else {
17                 printf("%dx", poly[0][i], poly[1][i]);
18             }
19         }
20     }
21 }
22
23 int** addPolynomial(int** poly1, int** poly2, int terms1, int terms2){
24     int size = terms1 + terms2;
25     int** sumPoly = malloc(sizeof(int)*2);
26     for(int i=0; i<2; i++){
27         sumPoly[i] = malloc(sizeof(int)*size);
28     }
29
30     int i=0, j=0, k=0;
31
32     while(i<terms1 && j<terms2){
33         if(poly1[1][i] == poly2[1][j]){
34             sumPoly[0][k] = poly1[0][i]+poly2[0][j];
35             sumPoly[1][k] = poly1[1][i];
36             i++;
37             j++;
38             k++;
39         } else if(poly1[1][i]>poly2[1][j]){
40             sumPoly[0][k] = poly1[0][i];
41             sumPoly[1][k] = poly1[1][i];
42             i++;
43             k++;
44         } else {
45             sumPoly[0][k] = poly2[0][j];
46             sumPoly[1][k] = poly2[1][j];
47             j++;
48             k++;
49         }
50     }
51
52     while(i<terms1){
53         sumPoly[0][k] = poly1[0][i];
54         sumPoly[1][k] = poly1[1][i];
55         i++;
56         k++;
57     }
58
59     while(j<terms2){
60         sumPoly[0][k] = poly2[0][j];
61         sumPoly[1][k] = poly2[1][j];
62         j++;
63         k++;
64     }
65
66     sumPoly[0] = realloc(sumPoly[0], sizeof(int)*k);
67     sumPoly[1] = realloc(sumPoly[1], sizeof(int)*k);
68
69     printPolynomial(sumPoly, k);
70 }
71
72
73 int** acceptPolynomial(int terms){
74
75     int** poly = malloc(sizeof(int)*2);
76     for(int i=0; i<2; i++){
77         poly[i] = malloc(sizeof(int)*terms);
78     }
79
80     for(int i=0; i<terms; i++){
81         printf("enter coefficient: ");
82         scanf("%d", &poly[0][i]);
83         printf("enter exponents: ");
84         scanf("%d", &poly[1][i]);
85     }
86     return poly;
87 }
88
89 int main(){
90     int n, m, total;
91     printf("Number of polynomials you want to add: ");
92     scanf("%d", &total);
93
94     int*** storePoly = malloc(sizeof(int**)*total);
95     int* term = malloc(sizeof(int)*total);
96
97     for(int i=0; i<total; i++){
98         printf("For Polynomial %d:\n", i+1);
99         printf("No. of terms: ");
100         scanf("%d", &term[i]);
101         storePoly[i] = acceptPolynomial(term[i]);
102     }
103
104     printf("The polynomials you entered are: \n");
105     for(int i=0; i<total; i++){
106         printPolynomial(storePoly[i], term[i]);
107         printf("\n");
108     }
109
110     int** sumPoly = addPolynomial(storePoly[0], storePoly[1], term[0], term[1]);
111
112 }
```

Number of polynomials you want to add: 2

For Polynomial 1:

No. of terms: 3

enter coefficient: 3

enter exponents: 3

enter coefficient: 2

enter exponents: 2

enter coefficient: 1

enter exponents: 0

For Polynomial 2:

No. of terms: 3

enter coefficient: 3

enter exponents: 3

enter coefficient: 2

enter exponents: 2

enter coefficient: 1

enter exponents: 0

The polynomials you entered are:

$$f(x) = 3x^3 + 2x^2 + 1$$

$$f(x) = 3x^3 + 2x^2 + 1$$

$$f(x) = 6x^3 + 4x^2 + 2$$

MULTIPLICATION OF POLYNOMIALS

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void printPolynomial(int** poly, int terms) {
5     printf("(x) = ");
6     for(int i = 0; i < terms; i++) {
7         if(i > 0 && poly[0][i] > 0) {
8             printf(" + ");
9         }
10
11         if(poly[0][i] != 0) {
12             if(poly[1][i] == 0) {
13                 printf("%d", poly[0][i]);
14             } else {
15                 printf("%dx^%d", poly[0][i], poly[1][i]);
16             }
17         }
18     }
19     printf("\n");
20 }
21
22 int** multiplyPolynomial(int** poly1, int** poly2, int terms1, int terms2, int* resultTerms) {
23     int maxTerms = terms1 + terms2;
24     int** prodPoly = malloc(sizeof(int*) * 2);
25     for(int i = 0; i < 2; i++) {
26         prodPoly[i] = calloc(maxTerms, sizeof(int));
27     }
28
29     int k = 0;
30
31     for(int i = 0; i < terms1; i++) {
32         for(int j = 0; j < terms2; j++) {
33             int coeff = poly1[0][i] * poly2[0][j];
34             int exp = poly1[1][i] + poly2[1][j];
35
36             int found = 0;
37             for(int l = 0; l < k; l++) {
38                 if(prodPoly[1][l] == exp) {
39                     prodPoly[0][l] += coeff;
40                     found = 1;
41                     break;
42                 }
43             }
44
45             if(!found) {
46                 prodPoly[0][k] = coeff;
47                 prodPoly[1][k] = exp;
48                 k++;
49             }
50         }
51     }
52
53     *resultTerms = k;
54     prodPoly[0] = realloc(prodPoly[0], sizeof(int) * k);
55     prodPoly[1] = realloc(prodPoly[1], sizeof(int) * k);
56
57     return prodPoly;
58 }
59
60 int** acceptPolynomial(int terms) {
61     int** poly = malloc(sizeof(int*) * 2);
62     for(int i = 0; i < 2; i++) {
63         poly[i] = malloc(sizeof(int) * terms);
64     }
65
66     for(int i = 0; i < terms; i++) {
67         printf("Enter coefficient: ");
68         scanf("%d", &poly[0][i]);
69         printf("Enter exponent: ");
70         scanf("%d", &poly[1][i]);
71     }
72     return poly;
73 }
74
75 int main() {
76     int n, m, total;
77     printf("Number of polynomials you want to multiply: ");
78     scanf("%d", &total);
79
80     int*** storePoly = malloc(sizeof(int**) * total);
81     int* term = malloc(sizeof(int) * total);
82
83     for(int i = 0; i < total; i++) {
84         printf("For Polynomial %d:\n", i + 1);
85         printf("No. of terms: ");
86         scanf("%d", &term[i]);
87         storePoly[i] = acceptPolynomial(term[i]);
88     }
89
90     printf("The polynomials you entered are: \n");
91     for(int i = 0; i < total; i++) {
92         printPolynomial(storePoly[i], term[i]);
93         printf("\n");
94     }
95
96     int resultTerms;
97     int** prodPoly = multiplyPolynomial(storePoly[0], storePoly[1], term[0], term[1], &resultTerms);
98
99     for(int i = 2; i < total; i++) {
100         int newTerms;
101         prodPoly = multiplyPolynomial(prodPoly, storePoly[i], resultTerms, term[i], &newTerms);
102         resultTerms = newTerms;
103     }
104
105     printf("The product of the polynomials is:\n");
106     printPolynomial(prodPoly, resultTerms);
107
108     // Free allocated memory
109     for(int i = 0; i < total; i++) {
110         for(int j = 0; j < 2; j++) {
111             free(storePoly[i][j]);
112         }
113         free(storePoly[i]);
114     }
115     free(storePoly);
116     free(term);
117
118     for(int i = 0; i < 2; i++) {
119         free(prodPoly[i]);
120     }
121     free(prodPoly);
122
123     return 0;
124 }
125
```

```
•$ ./multiply
Number of polynomials you want to multiply: 2
For Polynomial 1:
No. of terms: 3
Enter coefficient: 3
Enter exponent: 3
Enter coefficient: 2
Enter exponent: 2
Enter coefficient: 1
Enter exponent: 0
For Polynomial 2:
No. of terms: 3
Enter coefficient: 3
Enter exponent: 3
Enter coefficient: 2
Enter exponent: 2
Enter coefficient: 1
Enter exponent: 0
The polynomials you entered are:
f(x) = 3x^3 + 2x^2 + 1

f(x) = 3x^3 + 2x^2 + 1

The product of the polynomials is:
f(x) = 9x^6 + 12x^5 + 6x^3 + 4x^4 + 4x^2 + 1
```

IMPLEMENT SPARSE MATRIX

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void printArray(int** array, int rows, int columns){
5     for(int i=0; i<rows; i++){
6         for(int j=0; j<columns; j++){
7             printf("%d ", array[i][j]);
8         }
9         printf("\n");
10    }
11 }
12
13 int** createArray(int rows, int columns){
14
15     int** array = malloc(sizeof(int*)*rows);
16     for(int i=0; i<rows; i++){
17         array[i] = malloc(sizeof(int)*columns);
18     }
19
20     printf("Enter elements: \n");
21     for(int i=0; i<rows; i++){
22         for(int j=0; j<columns; j++){
23             printf("matrix[%d][%d] = ", i, j);
24             scanf("%d", &array[i][j]);
25         }
26     }
27
28     return array;
29 }
30
31 void printSparse(int** sparse){
32     printf("i j v\n");
33     for(int i=0; i<=sparse[0][2]; i++){
34         for(int j=0; j<=sparse[0][1]; j++){
35             printf("%d ", sparse[i][j]);
36         }
37         printf("\n");
38     }
39 }
40
41 int** convertToSparse(int** array, int rows, int columns){
42     int** sparse = malloc(sizeof(int*)*(rows+1));
43     for(int i=0; i<=rows; i++){
44         sparse[i] = malloc(sizeof(int)*columns);
45     }
46
47     int k=1;
48
49     for(int i=0; i<rows; i++){
50         for(int j=0; j<columns; j++){
51             if(array[i][j]!=0){
52                 sparse[k][0] = i;
53                 sparse[k][1] = j;
54                 sparse[k][2] = array[i][j];
55                 k++;
56             }
57         }
58     }
59     sparse[0][0] = rows;
60     sparse[0][1] = columns;
61     sparse[0][2] = k-1;
62     return sparse;
63 }
64
65 int main(){
66     int** array = createArray(3,3);
67     printf("Matrix is: \n");
68     printArray(array,3,3);
69     int** sparse = convertToSparse(array, 3,3);
70
71     printf("Matrix in Sparse Format: \n");
72     printSparse(sparse);
73 }
```

```
$ ./sm
Enter elements:
matrix[0][0] = 0
matrix[0][1] = 0
matrix[0][2] = 1
matrix[1][0] = 0
matrix[1][1] = 2
matrix[1][2] = 0
matrix[2][0] = 3
matrix[2][1] = 0
matrix[2][2] = 0
Matrix is:
0 0 1
0 2 0
3 0 0
Matrix in Sparse Format:
i j v
3 3 3
0 2 1
1 1 2
2 0 3
```