

DATA STRUCTURES LABORATORY

Course Code: 1BCSL306

VTU 2025 Scheme – Semester 3

EXPERIMENT – 05

Polynomial Addition

Using Singly Circular Linked List

SCLL with Header Node

**Simple Code • Easy to Remember • Easy to Write
in Exam**

SEARCH CREATORS

Your VTU Study Partner

Experiment 05

1. Aim

Aim

To develop a C program to represent and add two polynomials using a Singly Circular Linked List with a header node.

2. VTU Prescribed Experiment

VTU Prescribed Experiment

Develop a C program using a **Singly Circular Linked List (SCLL)** with header nodes for the following:

- Read the polynomial information such as number of terms, coefficient and power of each term.
- Represent a polynomial in one variable using a Singly Circular Linked List based on its degree.
- Add two polynomials **POLY1** and **POLY2** and store the result in **POLYSUM**.
- Use appropriate functions to create, display and add the polynomials.

3. Simple Theory

Polynomial:

A polynomial is an expression containing coefficients and powers of a variable.

Example:

$$P(x) = 5x^3 + 4x^2 + 2x + 7$$

Each term contains:

$$\boxed{\text{Coefficient} + \text{Power}}$$

For example:

$$5x^3$$

has:

$$\text{Coefficient} = 5$$

and:

$$Power = 3$$

4. Singly Circular Linked List

In a Singly Circular Linked List, the last node points back to the header node.

Each polynomial node stores:

$$Coefficient + Power + Next$$

Node representation:

```

1 struct Node
2 {
3     int coef;
4     int pow;
5     struct Node *next;
6 };

```

4.1. Header Node

A header node is a special first node.

Initially:

$$head \rightarrow next = head$$

Therefore, an empty circular list looks like:

$$HEAD$$

After inserting polynomial terms:

$$HEAD \rightarrow Term_1 \rightarrow Term_2 \rightarrow Term_3 \rightarrow HEAD$$

5. Polynomial Representation

Consider:

$$P(x) = 5x^3 + 4x^2 + 2$$

The linked list representation is:

$$HEAD \rightarrow (5, 3) \rightarrow (4, 2) \rightarrow (2, 0) \rightarrow HEAD$$

where:

$$(5, 3)$$

means:

$$\text{Coefficient} = 5, \quad \text{Power} = 3$$

6. Polynomial Addition Logic

Consider:

$$P_1(x) = 5x^3 + 4x^2 + 2$$

and:

$$P_2(x) = 3x^3 + 2x + 1$$

Compare powers of both polynomials.

If powers are equal:

$$\boxed{\text{Coefficient} = \text{Coef}_1 + \text{Coef}_2}$$

For example:

$$5x^3 + 3x^3 = 8x^3$$

Therefore:

$$P_{SUM}(x) = 8x^3 + 4x^2 + 2x + 3$$

7. Algorithm

7.1. Creating a Polynomial

1. Start.
2. Create a header node.
3. Make header point to itself.
4. Read number of polynomial terms.
5. For each term:
 - Read coefficient.

- Read power.
 - Create a new node.
 - Insert the node before the header.
6. Return the header node.

7.2. Displaying a Polynomial

1. Start from the first node after the header.
2. Display coefficient and power.
3. Move to the next node.
4. Repeat until the header node is reached again.

7.3. Adding Two Polynomials

1. Take pointers to the first nodes of POLY1 and POLY2.
2. Compare their powers.
3. If powers are equal:
 - Add coefficients.
 - Store result in POLYSUM.
 - Move both pointers.
4. If power of POLY1 is greater:
 - Copy POLY1 term to POLYSUM.
 - Move POLY1 pointer.
5. Otherwise:
 - Copy POLY2 term to POLYSUM.
 - Move POLY2 pointer.
6. Copy any remaining terms.
7. Stop.

Easy Exam Memory

Each node:

Coefficient, Power, Next

Circular condition:

Last Node → Header

Polynomial addition:

Same Power → Add Coefficients

Different powers:

Copy Higher Power

8. C Program – Easy Student Version

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 struct Node
5 {
6     int coef;
7     int pow;
8     struct Node *next;
9 };
10
11 struct Node* createHeader()
12 {
13     struct Node *head;
14
15     head=(struct Node*)malloc(sizeof(struct Node));
16     head->next=head;
17
18     return head;
19 }
20
21 void insert(struct Node *head, int c, int p)
22 {
23     struct Node *newnode, *temp;
24
25     newnode=(struct Node*)malloc(sizeof(struct Node));
26
27     newnode->coef=c;
28     newnode->pow=p;
29
30     temp=head;
31
32     while(temp->next!=head)
33         temp=temp->next;
34
35     newnode->next=head;
36     temp->next=newnode;
37 }
38
39 struct Node* createPoly()
40 {
41     struct Node *head;
42     int n,i,c,p;
43
44     head=createHeader();
45
46     printf("Enter number of terms: ");
47     scanf("%d",&n);
48
49     printf("Enter terms in descending order of power\n");
50
```

```
51     for(i=0;i<n;i++)
52     {
53         printf("Enter coefficient and power: ");
54         scanf("%d%d",&c,&p);
55
56         insert(head,c,p);
57     }
58
59     return head;
60 }
61
62 void display(struct Node *head)
63 {
64     struct Node *temp;
65
66     temp=head->next;
67
68     if(temp==head)
69     {
70         printf("Polynomial is Empty\n");
71         return;
72     }
73
74     while(temp!=head)
75     {
76         printf("%dx^%d",temp->coef,temp->pow);
77
78         temp=temp->next;
79
80         if(temp!=head)
81             printf(" + ");
82     }
83
84     printf("\n");
85 }
86
87 struct Node* addPoly(struct Node *p1, struct Node *p2)
88 {
89     struct Node *sum;
90     struct Node *a, *b;
91     int c;
92
93     sum=createHeader();
94
95     a=p1->next;
96     b=p2->next;
97
98     while(a!=p1 && b!=p2)
99     {
100         if(a->pow==b->pow)
101         {
102             c=a->coef+b->coef;
103
```

```
104         if(c!=0)
105             insert(sum,c,a->pow);
106
107         a=a->next;
108         b=b->next;
109     }
110     else if(a->pow>b->pow)
111     {
112         insert(sum,a->coef,a->pow);
113         a=a->next;
114     }
115     else
116     {
117         insert(sum,b->coef,b->pow);
118         b=b->next;
119     }
120 }
121
122 while(a!=p1)
123 {
124     insert(sum,a->coef,a->pow);
125     a=a->next;
126 }
127
128 while(b!=p2)
129 {
130     insert(sum,b->coef,b->pow);
131     b=b->next;
132 }
133
134 return sum;
135 }
136
137 int main()
138 {
139     struct Node *poly1, *poly2, *polysum;
140
141     printf("Enter POLY1\n");
142     poly1=createPoly();
143
144     printf("\nEnter POLY2\n");
145     poly2=createPoly();
146
147     printf("\nPOLY1 = ");
148     display(poly1);
149
150     printf("POLY2 = ");
151     display(poly2);
152
153     polysum=addPoly(poly1,poly2);
154
155     printf("POLYSUM = ");
156     display(polysum);
```

```
157  
158     return 0;  
159 }
```

9. Simple Program Explanation

9.1. Node Structure

```
1 struct Node  
2 {  
3     int coef;  
4     int pow;  
5     struct Node *next;  
6 };
```

Each node contains:

- Coefficient
- Power
- Address of next node

9.2. createHeader()

The header node is created using:

```
1 head=(struct Node*)malloc(sizeof(struct Node));  
2 head->next=head;
```

The statement:

$$head \rightarrow next = head$$

makes the list circular.

9.3. insert() Function

The function moves until:

```
1 temp->next==head
```

Then the new node is inserted before the header node.

Therefore:

$$Last\ Node \rightarrow Header$$

is maintained.

9.4. createPoly() Function

This function reads:

Number of Terms

and for every term reads:

Coefficient + Power

For easy polynomial addition, terms are entered in:

Descending Order of Power

9.5. display() Function

Traversal begins from:

1 `head->next`

and continues until:

1 `temp==head`

This is the main circular linked-list condition.

9.6. addPoly() Function

Three simple cases are used.

Case 1: Powers are equal

$$a \rightarrow pow = b \rightarrow pow$$

Then:

$$c = a \rightarrow coef + b \rightarrow coef$$

Case 2: POLY1 power is greater

Copy the POLY1 term.

Case 3: POLY2 power is greater

Copy the POLY2 term.

Easy Exam Memory

Remember the program in this order:

Node → Header → Insert → Create → Display → Add → Main

Most important comparison:

$a->pow == b->pow$

then:

Add Coefficients

If powers are different:

Copy the term having higher power

10. Sample Input and Output

Let:

$$POLY1 = 5x^3 + 4x^2 + 2$$

and:

$$POLY2 = 3x^3 + 2x + 1$$

Input:

```
1
2 Enter POLY1
3
4 Enter number of terms: 3
5
6 Enter terms in descending order of power
7
8 Enter coefficient and power: 5 3
9 Enter coefficient and power: 4 2
10 Enter coefficient and power: 2 0
11
12
13 Enter POLY2
14
15 Enter number of terms: 3
16
17 Enter terms in descending order of power
18
19 Enter coefficient and power: 3 3
20 Enter coefficient and power: 2 1
21 Enter coefficient and power: 1 0
```

Output:

```
1
2 POLY1 = 5x^3 + 4x^2 + 2x^0
3
4 POLY2 = 3x^3 + 2x^1 + 1x^0
5
6 POLYSUM = 8x^3 + 4x^2 + 2x^1 + 3x^0
```

Therefore:

$$POLYSUM = 8x^3 + 4x^2 + 2x + 3$$

11. How Addition Works

$$POLY1 = 5x^3 + 4x^2 + 2$$

$$POLY2 = 3x^3 + 2x + 1$$

Compare first terms:

$$5x^3 + 3x^3$$

Same power:

$$\boxed{8x^3}$$

Next:

$$4x^2$$

POLY2 has no x^2 term.

Therefore copy:

$$\boxed{4x^2}$$

Next:

$$2x$$

comes from POLY2:

$$\boxed{2x}$$

Finally:

$$2 + 1$$

gives:

$$\boxed{3}$$

Hence:

$$\boxed{8x^3 + 4x^2 + 2x + 3}$$

12. Simple Test Cases

Test	Input / Operation	Expected Result
1	Create one polynomial	Polynomial should be represented using SCLL.
2	Display POLY1	All terms should be displayed in descending power order.
3	Add equal powers	Coefficients should be added.
4	Add different powers	Higher-power term should be copied to result.
5	Add POLY1 and POLY2	Correct POLYSUM should be displayed.

13. Result

Result

Thus, the C program to represent and add two polynomials using a **Singly Circular Linked List with a header node** was successfully implemented.

The program performs:

- Creation of POLY1
- Creation of POLY2
- Polynomial Representation
- Polynomial Display
- Addition of POLY1 and POLY2
- Creation of POLYSUM

14. Important Viva Questions

Important Viva Questions

1. What is a linked list?

A linked list is a collection of nodes connected using pointers.

2. What is a Singly Circular Linked List?

It is a linked list in which the last node points back to the first or header node.

3. What is a header node?

A header node is a special node placed at the beginning of the linked list.

4. What does each polynomial node contain?

Coefficient, Power, Next

5. What makes this linked list circular?

The last node points to the header node.

6. What is the initial condition of the header node?

$head \rightarrow next = head$

7. How are polynomial terms represented?

Using coefficient and power.

8. What happens when two terms have the same power?

Their coefficients are added.

9. What happens when powers are different?

The term having the higher power is copied to the result first.

10. Why are terms entered in descending order of power?

It makes polynomial addition easier.

11. Which function is used to allocate memory dynamically?

`malloc()`

12. Which header file contains malloc()?

`stdlib.h`

13. What is POLYSUM?

POLYSUM is the linked list containing the sum of POLY1 and POLY2.

14. What is the advantage of a linked list for polynomials?

Only required polynomial terms are stored dynamically.

15. How do we know traversal of an SCLL is complete?

Traversal is complete when the pointer reaches the header node again.

15. 1-Minute Exam Revision

Easy Exam Memory

Node

$$\text{Coefficient} + \text{Power} + \text{Next}$$

Circular Condition

$$\text{head} \rightarrow \text{next} = \text{head}$$

for an empty list.

Traversal Stop

$$\text{temp} == \text{head}$$

Same Powers

$$\text{Coef}_{sum} = \text{Coef}_1 + \text{Coef}_2$$

Different Powers

$$\text{Copy Higher Power}$$

Functions

$$\text{createHeader}()$$
$$\text{insert}()$$
$$\text{createPoly}()$$
$$\text{display}()$$
$$\text{addPoly}()$$

16. Easy Code Memory Map

Remember

Remember:

struct Node

contains:

coef, pow, next

Then:

Header → Insert → Create → Display → Add

The main addition logic is only:

Same Power → Add

Greater Power → Copy

Remaining Terms → Copy

Most important SCLL condition:

Pointer reaches Header ⇒ Stop

SEARCH CREATORS

1BCSL306 – Data Structures Laboratory

EXPERIMENT 05 COMPLETED

Polynomial Addition

Singly Circular Linked List with Header Node

Understand → Remember → Write → Execute →
Viva

Your VTU Study Partner