14

Case Study of Data Structure (link list)

Case 1: Address Book

- Problem definition
  - An useful address book
    - Name
    - Phone Number
    - E-mail address
  - It includes two functions
    - Print address book (File processing)
    - Edit address book (Linked lists)

Source Code (1/4)

```
#include <iostream>
#include <fstream>

using namespace std;

void newdata(); // Key in new address data
void output(); // Output the address data to a file

struct node{
    char name[20];
    char phone_number[20];
    char e_mail[20];
    node *nxt;
};
node *start_ptr=NULL;
```

Source Code (2/4)

```
int main(){

    int select;

    do{
        select = 0; //Initialization
        cout << "Please key in..." << endl
             << "1. New address data" << endl
             << "2. Output the address data to a file" << endl
             << "3. Exit the program" << endl;
        cin >> select;

        if(select == 1) //key in new address data
            newdata();
        if(select ==2) //output the address data to a file
            output();
        cout << endl;

    }while(select != 3);
    return 0;
}
```

Source Code (3/4)

```
void output(){

    ofstream file("Address Book.txt", ios::app); // Output Address Book
    node *temp;

    temp = start_ptr;
    do{
        if(temp == NULL)
            file << "End of the list" << endl;
        else{
            file << "Name: " << temp->name << endl;
            file << "Phone Number: " << temp->phone_number << endl;
            file << "e-mail: " << temp->e_mail << endl << endl;
            temp=temp->nxt;
        }
    }while(temp!=NULL);
}
```

Source Code (4/4)

```
void newdata(){

    node *temp, *temp2;

    temp = new node;
    temp2=start_ptr;
    cout << "Please enter the name of the person: "; //key in name
    cin >> temp-> name;
    cout << "Please enter the phone number of the person: "; //key in phone number
    cin >> temp-> phone_number;
    cout << "Please enter the e-mail of the person: "; // key in e-mail
    cin >> temp->e_mail;
    temp -> nxt = NULL;

    if(start_ptr==NULL)
        start_ptr=temp;
    else{
        temp2=start_ptr;
        while(temp2->nxt != NULL)
            temp2=temp2->nxt;
        temp2->nxt=temp;
    }
}
```

Case 2: Polynomial Addition

■Problem Definition

- 多項式加法 **(polynomial addition)**
- 以 **Linked List** 儲存、處理資料。
- 使用者可自行輸入有多少項次、係數及指數的值，並且將加式與被加式相加，將其結果以降冪輸出。

■Data Structure

```
typedef struct poly *poly_pointer;
typedef struct poly {
 int           exp;          // exp
 int           coeff;        //coefficient
 poly_pointer  next;         //下一項

};
```

Source Code (1/5)

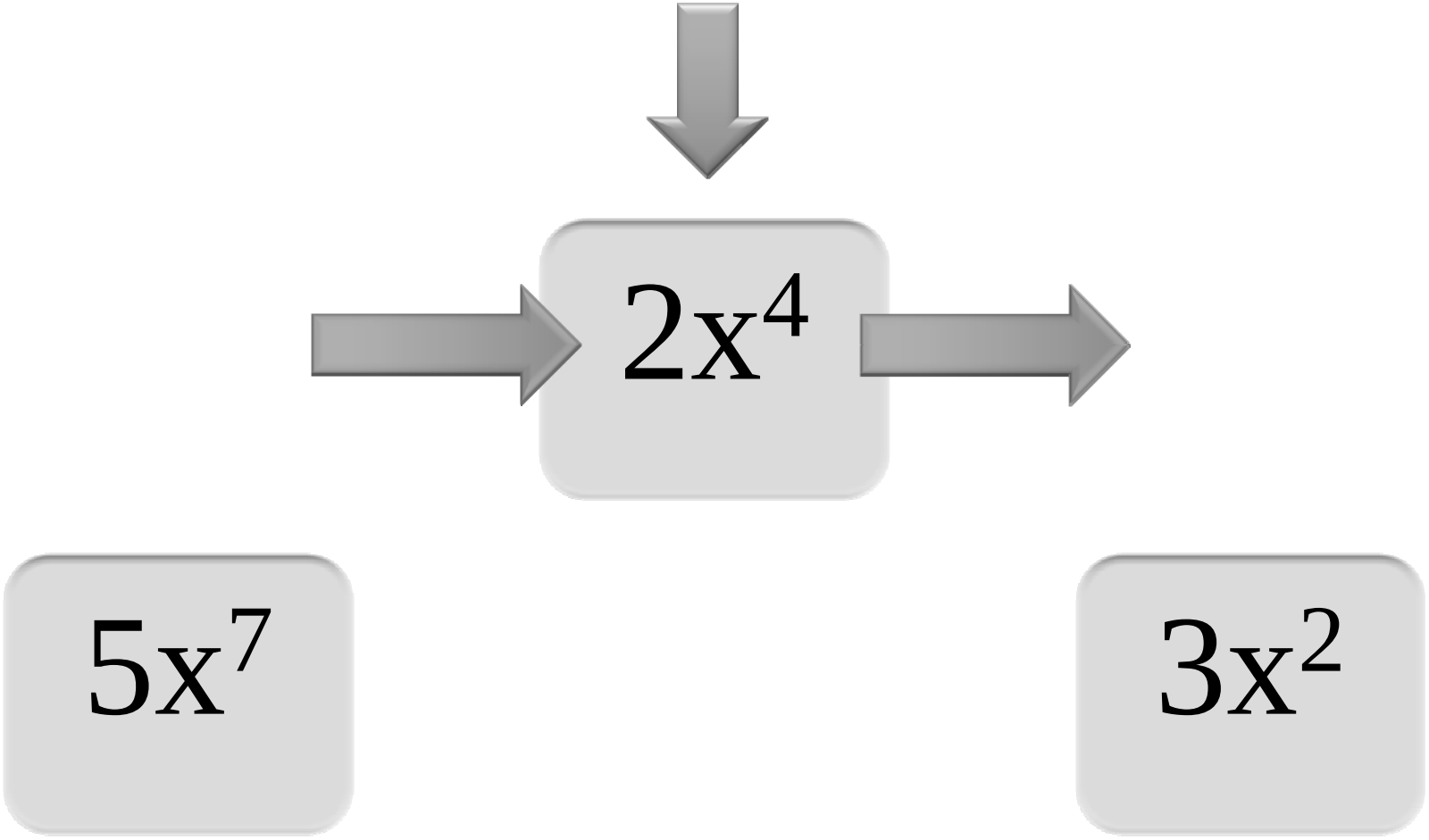
■Main function

```
int main()
{
 poly_pointer   poly_addend,poly_summend; // addend summend
 poly_addend         = (poly_pointer)malloc(sizeof(poly));
 poly_summend        = (poly_pointer)malloc(sizeof(poly));
 poly_addend ->coeff=0;
 poly_summend ->coeff=0;
 poly_addend ->next = NULL;
 poly_summend->next = NULL;

 printf("**********************\n* polynamial adder *\n**********************\n\n");
 printf("*******addend*******\n");
 down(&poly_addend);
 printf("\n\n*******summend*******\n");
 down(&poly_summend);

 adder(poly_addend,poly_summend);
 system("PAUSE");
 return EXIT_SUCCESS;
}
```

Illustration of 降冪排列



Source Code (2/5)

■降冪排列:down (1/2)

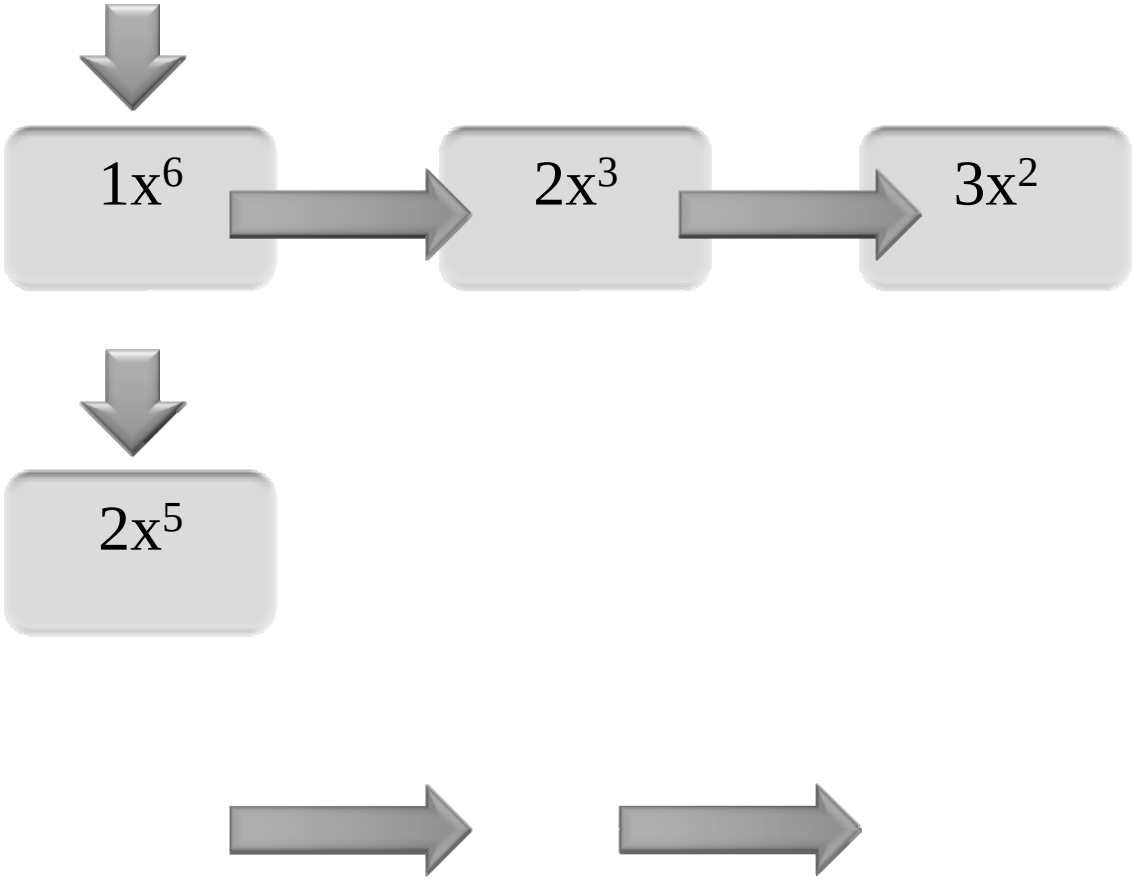
```
printf("how many terms:");                                  //輸入爲幾次項
scanf("%d",&i);
   for(j=0;j<i;j++){
      poly_node       = (poly_pointer)malloc(sizeof(poly));
      poly_node->next = NULL;
      poly_node->exp = exp;
      poly_node->coeff = coeff;
      printf("\nPlease input exp of polynomial:");     //輸入次方
      scanf("%d",&poly_node->exp);
      printf("Please input coeff of polynomial:");     // 輸入係數
      scanf("%d",&poly_node->coeff);
      if((*poly_adder)->next==NULL) (*poly_adder)->next = poly_node;
      done=0;                                                  //未完成 done=0
      poly_temp = *(poly_adder);
```

Source Code (3/5)

■降冪排列:down （ 2/2 ）

```
while(!done && j!=0){
      if(poly_node->exp > (*poly_adder)->next->exp){ //輸入次方>第一項
         poly_node->next = (*poly_adder)->next;      //輸入取代爲第一項
         (*poly_adder)->next   = poly_node;
         done=1;                                     //完成  done=1
         }
      if(poly_temp->exp == poly_node->exp){          //輸入的次方相同  係數相加
         poly_temp->coeff = poly_temp->coeff + poly_node->coeff ;
         free (poly_node); done=1;                  /
         }
      else{
           if(poly_temp->next == NULL){              //NEXT==NULL  直接輸出
              poly_temp->next = poly_node; done=1;
              }
           else if(poly_temp->next->exp < poly_node->exp){  //若下一項次方較小
                   poly_node->next = poly_temp->next;
                   poly_temp->next = poly_node;
                   done=1;  ;
                   }
           }
      poly_temp = poly_temp->next;                  //下一個項次
      }
   }
(*poly_adder) = (*poly_adder)->next;
```

Illustration of Polynomial Addition



Source Code (4/5)

■Adder function

```
void adder(poly_pointer addend, poly_pointer summend){
 poly_pointer poly_answer;
 poly_pointer rear;
 poly_answer = (poly_pointer)malloc(sizeof(poly));
 rear = (poly_pointer)malloc(sizeof(poly));
 poly_answer = rear;

 while(addend && summend){
      if(summend->exp < addend->exp){                  //若addend次方較高  先輸出addend項次
        if(addend->coeff!=0)
        attach(addend->coeff,addend->exp,&rear);
        addend = addend->next;                         // call attach function
        }
      else if (summend->exp > addend->exp){            //若summend次方較高  先輸出summend項次
        if(summend->coeff!=0)
        attach(summend->coeff,summend->exp,&rear);     // call attach function
        summend = summend->next;
        }
      else if (summend->exp = addend->exp){            //若summend次方較高  將係數相加
         if(summend->coeff+addend->coeff!=0)
        attach(addend->coeff+summend->coeff,addend->exp,&rear);
        summend = summend->next;
        addend  = addend->next;
        }
 }                                                     //copy剩下的項次
 for (;addend ; addend  = addend->next ) attach(addend->coeff ,  addend->exp, &rear);
 for (;summend; summend = summend->next) attach(summend->coeff, summend->exp, &rear);

 rear->next  = NULL;
 poly_answer = poly_answer->next;
```

Source Code (5/5)

■Attach function

```
void attach(int coeff,int exp,poly_pointer *rear){
 poly_pointer node;
 node          = (poly_pointer)malloc(sizeof(poly));   //  將node指向宣告的記憶體
 node->coeff   = coeff;                                //  copy coeff
 node->exp     = exp;                                  //  copy exp
 (*rear)->next = node;                                 //  將 rear->NEXT 指到新的node
  *rear        = node;                                 //  rear指向 node
 }
```

Result on the Screen