Polynomial eval using SLL

27 July 2024 21:26

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef struct node* nodeptr;
struct node{
   int power;
   int coef;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr position;
LIST createList(void){
   LIST L = (LIST)malloc(sizeof(struct node));
   if(L==NULL) printf("fatal error");
   else{
      L->next=NULL;
      return L;
   }
}

int isLast(position P,LIST L){
   return(P->next==NULL);
}

void insert(int power,int coef,LIST L,position P){
   position tmp_cell = (position) malloc(sizeof(struct node));
   if(tmp_cell==NULL)printf("fatal error");
   else{
      tmp_cell->power = power;
      tmp_cell->coef = coef;
      tmp_cell->next=P->next;
      P->next=tmp_cell;
   }
}

int calculatePolynomial(LIST L,position P,int x){
   int ans=0;
   P=L->next;
   while(P!=NULL){
      ans+=(P->coef)*pow(x,P->power);
      P=P->next;
   }
   return ans;
}

int main(){
   LIST L = createList();
   position P = L;
```

```
    int deg ; scanf("%d",&deg);
    for(int i=deg;i>=0;i--){
        int temp_cof ; scanf("%d",&temp_cof);
        insert(i,temp_cof,L,P);
    }
    int num ; scanf("%d",&num);
    printf("%d",calculatePolynomial(L,P,num));
}
```

Bubblesort in LL

13 September 2024 14:55

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct node* nodeptr;
struct node{
   int element;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr POSITION;

LIST createList(void){
   LIST L = (LIST) malloc(sizeof(struct node));
   if(L==NULL) printf("fatal error");
   else{
      L->next=NULL;
   }
   return L;
}
void insertElement(LIST L , POSITION P,int x){
   POSITION tmp_cell = (POSITION)malloc(sizeof(struct node));
   if(tmp_cell==NULL) printf("Fatal err");
   else{
      tmp_cell->element=x;
      tmp_cell->next=NULL;
      POSITION current = L;
      while(current->next!=NULL)
         current = current->next;
      current->next=tmp_cell;
   }
}
void bubbleSort(LIST L,POSITION P,int n){
   if(L==NULL || L->next==NULL) return;
   int swapped;
   int i,j;
   for(i=0;i<n;i++){
      POSITION current = L->next;
      POSITION nextNode = current->next;
      for(j=0;j<n-i-1;j++){
         if(current->element > current->next->element){
            int temp = current->element;
            current->element = current->next->element;
            current->next->element = temp;
            swapped =1;
         }
         current = current->next;
         nextNode = current->next;
      }
      if(!swapped) break;
   }
}
```

```
void displayElements(LIST L,POSITION P){
   POSITION current= L->next;
   while(current!=NULL){
      printf("%d ",current->element);
      current = current->next;
   }
}

int main(){
   int n ; scanf("%d",&n);
   LIST L = createList();
   POSITION P = L;
   for(int i=0;i<n;i++){
      int temp; scanf("%d",&temp);
      insertElement(L,P,temp);
   }
   bubbleSort(L,P,n);
   displayElements(L,P);
}
```

Delete last node

13 September 2024 15:20

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct node* nodeptr;

struct node{
   int element;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr POSITION;

LIST createList(void){
   LIST L = (LIST)malloc(sizeof(struct node));
   if(L==NULL) printf("Fatal err");
   else{
      L->next=NULL;
   }
   return L;
}

void insertElement(LIST L,int x){
   POSITION tmp = (POSITION) malloc(sizeof(struct node));
   if(tmp==NULL) printf("Fatal err");
   else{
      tmp->element=x;
      tmp->next = NULL;
      POSITION current = L;
      while(current->next!=NULL){
         current = current->next;
      }
      current->next=tmp;
   }
}

void deleteLastNode(LIST L){
   POSITION current = L;
   POSITION prev=NULL;
   while(current->next!=NULL){
      prev = current;
      current = current->next;
   }
   if(prev!=NULL){
      prev->next=NULL;
   }
   free(current);
}
void displayElements(LIST L){
   POSITION current = L->next;
   while(current!=NULL){
```

```
        printf("%d ",current->element);
        current = current->next;
    }
}
int main(){
    int n; scanf("%d",&n);
    LIST L = createList();
    for(int i=0;i<n;i++){
        int tmp ; scanf("%d",&tmp);
        insertElement(L,tmp);
    }
    deleteLastNode(L);
    displayElements(L);
}
```

Insert elements

13 September 2024 15:20

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct node* nodeptr;

struct node{
   int element;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr POSITION;

LIST createList(){
   LIST L = (LIST)malloc(sizeof(struct node));
   if(L==NULL) printf("Fatal err");
   else{
      L->next=NULL;
   }
   return L;
}

void insertElement(LIST L,int x){
   POSITION tmp_cell = (POSITION)malloc(sizeof(struct node));
   if(tmp_cell==NULL) printf("Fatal err");
   else{
      tmp_cell->element = x;
      tmp_cell->next=L->next;
      L->next=tmp_cell;
   }
}
void displayElements(LIST L){
   POSITION current = L->next;
   while(current!=NULL){
      printf("%d ",current->element);
      current = current->next;
   }
}

int main(){
   int n; scanf("%d",&n);
   LIST L = createList();
   for(int i=0;i<n;i++){
      int temp; scanf("%d",&temp);
      insertElement(L,temp);
   }
   displayElements(L);

}
```

Checking polynomials equality

13 September 2024 15:20

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct node* nodeptr;
struct node{
   int coef;
   int power;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr POSITION;

LIST createList(){
   LIST L = (LIST)malloc(sizeof(struct node));
   if(L==NULL) printf("fatal err");
   else{
      L->next=NULL;
      return L;
   }
}

void insertElement(LIST L,int coef,int power){
   POSITION tmp = (POSITION)malloc(sizeof(struct node));
   if(tmp==NULL) printf("fatal error");
   else{
      tmp->coef=coef;
      tmp->power=power;
      tmp->next=NULL;
      POSITION current = L;
      while(current->next!=NULL){
         current = current->next;
      }
      current->next=tmp;
   }
}

void displayPolynomial(LIST L , int deg){
   POSITION current = L->next;
   while(current!=NULL){
      if(current!=L->next){
         printf(" + ");
      }
      printf("(%dx^%d)",current->coef,current->power);
      current = current->next;
   }
}

int areEqual(LIST L1 , LIST L2){
   POSITION current1 = L1->next;
   POSITION current2 = L2->next;
   while(current1!=NULL && current2!=NULL){
```

```
        if(current1->coef!=current2->coef || current1->power != current2->power){
            return 0;
        }
        current1=current1->next;
        current2=current2->next;
    }
    if(current1!=NULL || current2!=NULL){
        return 0;
    }
    return 1;
}


int main(){
    int n1; scanf("%d",&n1);
    LIST L1 = createList();
    for(int i=0;i<n1;i++){
        int coef ; scanf("%d",&coef);
        int power; scanf("%d",&power);
        insertElement(L1,coef,power);
    }
    int n2 ; scanf("%d",&n2);
    LIST L2 = createList();
    for(int j=0;j<n2;j++){
        int coef ; scanf("%d",&coef);
        int power; scanf("%d",&power);
        insertElement(L2,coef,power);
    }
    printf("Polynomial 1: ");
    displayPolynomial(L1,n1);
    printf("\n");
    printf("Polynomial 2: ");
    displayPolynomial(L2,n2);
    int res = areEqual(L1,L2);
    printf("\n");
    if(res) printf("Polynomials are Equal.");
    else printf("Polynomials are Not Equal.");
}
```

reversedEven+ReversedOdd

13 September 2024 16:57

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct node* nodeptr;
struct node{
   int element;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr POSITION;

LIST createList(){
   LIST L = (LIST)malloc(sizeof(struct node));
   if(L==NULL) printf("fatal err");
   else{
      L->next=NULL;
   }
   return L;
}

void insertElements(LIST L,int x){
   POSITION cell = (POSITION)malloc(sizeof(struct node));
   if(cell==NULL) printf("fatal err");
   else{
      cell->element = x;
      cell->next=NULL;
      POSITION current = L;
      while(current->next !=NULL){
         current=current->next;
      }
      current->next=cell;
   }
}
void displayElements(LIST L){
   POSITION current = L->next;
   while(current!=NULL){
      printf("%d ",current->element);
      current = current->next;
   }
}
LIST reverse(LIST L){
   LIST L_rev = createList();
   POSITION current = L->next;
   while(current!=NULL){
      POSITION cell = (POSITION)malloc(sizeof(struct node));
      cell->element = current->element;
      cell->next = L_rev->next;
      L_rev->next=cell;
      current = current->next;
   }
   return L_rev;
```

```
}
LIST concatenate(LIST L1,LIST L2){
   POSITION current = L1->next;
   while(current->next!=NULL){
      current = current->next;
   }
   current->next=L2->next;
   return L1;
}
void findEvenOdd(LIST L,LIST L_even,LIST L_odd,int n){
   POSITION current = L->next;
   while(current!=NULL){
      if(current->element%2==0){
         insertElements(L_even,current->element);
      }else{
         insertElements(L_odd,current->element);
      }
      current = current->next;
   }

}
void reArrange(LIST L,int n){
   LIST L_even = createList();
   LIST L_odd = createList();
   findEvenOdd(L,L_even,L_odd,n);
   LIST L_rev_even = reverse(L_even);
   LIST L_rev_odd = reverse(L_odd);
   L = concatenate(L_rev_even,L_rev_odd);
   displayElements(L);
}


int main(){
   int n; scanf("%d",&n);
   LIST L = createList();
   for(int i=0;i<n;i++){
      int temp;scanf("%d",&temp);
      insertElements(L,temp);
   }
   reArrange(L,n);

}
```

Remove element from back

13 September 2024 17:51

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct node* nodeptr;
struct node{
   int element;
   nodeptr next;
};
typedef nodeptr LIST;
typedef nodeptr POSITION;

LIST createList(void){
   LIST L = (LIST) malloc(sizeof(struct node));
   if(L==NULL) printf("fatal error");
   else{
      L->next=NULL;
   }
   return L;
}

void addElement(LIST L,int x){
   POSITION cell = (POSITION)malloc(sizeof(struct node));
   if(cell==NULL) printf("fatal error");
   else{
      cell->element = x;
      cell->next=NULL;
      POSITION current = L;
      while(current->next!=NULL){
         current = current ->next;
      }
      current->next=cell;
   }
}
void displayElements(LIST L){
   POSITION current = L->next;
   while(current!=NULL){
      printf("%d ",current->element);
      current = current ->next;
   }
}
void removeElement(LIST L,int n,int pos){
   int posFromFront = n-pos+1;
   POSITION prev = L;
   POSITION current = L->next;
   int currentpost =1;
   while(currentpost<posFromFront && current !=NULL){
      prev = current;
      current = current->next;
      currentpost++;
   }
```

```
    if(current !=NULL){
        prev->next=current->next;
        free(current);
    }
}


int main(){
    int n; scanf("%d",&n);
    LIST L = createList();
    for(int i=0;i<n;i++){
        int temp; scanf("%d",&temp);
        addElement(L,temp);
    }
    int x; scanf("%d",&x);
    removeElement(L,n,x);
    displayElements(L);

}
```