|  |  |
| --- | --- |
| 1 | Create a polynomial using a linked list. |
|  | #pragma pack(1)  struct xxx  {  int cof;  int exp;  struct xxx \*ad;  };  struct xxx \*create();  main()  {  struct xxx \*p;  p=create();  traverse(p);  }  struct xxx \*create()  {  struct xxx \*p,\*q,\*r;  char ch[5];  p=malloc(sizeof(struct xxx));  printf("Enter coefficient:");  scanf("%d",&p->cof);  printf("Enter exponent:");  scanf("%d",&p->exp);  r=p;  while(1)  {  printf("Do u continue yes/no:");  scanf("%s",ch);  if(strcmp(ch,"no")==0)  break;  q=malloc(sizeof(struct xxx));  p->ad=q;  p=q;  printf("Enter coefficient:");  scanf("%d",&p->cof);  printf("Enter exponent:");  scanf("%d",&p->exp);  }  p->ad=0;  return r;  }  int traverse(struct xxx \*p)  {  while(p->ad!=0)  {  printf("%dx^%d + ",p->cof,p->exp);  p=p->ad;  }  printf("%dx^%d ",p->cof,p->exp);    } |
| 2 | Add between two polynomial using a linked list. |
|  | #pragma pack(1)  struct xxx  {  int cof;  int exp;  struct xxx \*ad;  };  struct xxx \*create();  struct xxx \*add(struct xxx \*p1,struct xxx \*p2);  main()  {  struct xxx \*p1,\*p2,\*p3;  p1=create();  traverse(p1);  printf("\n");  p2=create();  traverse(p2);    printf("\n");  p3=add(p1,p2);  sort(p3);  traverse(p3);  }  int sort(struct xxx \*p)  {  struct xxx \*q;  int t1;  while(p!=0)  {  q=p->ad;  while(q!=0)  {  if(p->exp < q->exp)  {  t1=p->exp;  p->exp=q->exp;  q->exp=t1;  t1=p->cof;  p->cof=q->cof;  q->cof=t1;  }  q=q->ad;  }  p=p->ad;  }  }  struct xxx \*add(struct xxx \*p1,struct xxx \*p2)  {  struct xxx \*p,\*r,\*s;  struct xxx \*q;  int f=0,flag=0;  while(p1!=0)  {  q=p2;  while(q!=0)  {  if(p1->exp==q->exp)  {  f=1;  q->cof=q->cof+p1->cof;  break;  }  q=q->ad;  }  if(f==0)  {  if(flag==0)  {  p=malloc(sizeof(struct xxx));  p->cof=p1->cof;  p->exp=p1->exp;  s=p;  flag=1;  }  else  {  r=malloc(sizeof(struct xxx));  p->ad=r;  p=r;  p->cof=p1->cof;  p->exp=p1->exp;  }  }  f=0;  p1=p1->ad;  }  p->ad=p2;  return s;  }  int traverse(struct xxx \*p)  {  while(p->ad!=0)  {  printf("%dx^%d + ",p->cof,p->exp);  p=p->ad;  }  printf("%dx^%d ",p->cof,p->exp);    }  struct xxx \*create()  {  struct xxx \*p,\*q,\*r;  char ch[5];  p=malloc(sizeof(struct xxx));  printf("Enter coefficient:");  scanf("%d",&p->cof);  printf("Enter exponent:");  scanf("%d",&p->exp);  r=p;  while(1)  {  printf("Do u continue yes/no:");  scanf("%s",ch);  if(strcmp(ch,"no")==0)  break;  q=malloc(sizeof(struct xxx));  p->ad=q;  p=q;  printf("Enter coefficient:");  scanf("%d",&p->cof);  printf("Enter exponent:");  scanf("%d",&p->exp);  }  p->ad=0;  return r;  } |
| 3 | Implement linear search in a linked list |
|  | int linear\_search(struct xxx \*p)  {  int n;  printf("Enter number to search:");  scanf("%d",&n);  while(p!=0)  {  if(n==p->roll)  {  printf("Found");  return;  }  p=p->ad;  }  printf("Not found");  } |
| 4 | Find the nth element from the last of linked list |
|  | int dele\_node(struct xxx \*p,int pos)  {  int i;  struct xxx \*r;  for(i=1;i<pos;i++)  {  p=p->ad;  }  while(p->ad!=0)  {  r=p;  p->roll=p->ad->roll;  p=p->ad;  }  r->ad=0;  free(p);  } |