بسم الله الرحمن الرحيم

هياكل البيانات بـ لغة السي

كل برنامج في ملف منفصل..
كل البرامج مجموعه في ملف واحد ...

- Stack
- Queue
- Conversion infix → postfix
- Conversion infix → prefix
- Conversion prefix → infix
- Conversion prefix → postfix
- Conversion postfix → infix
- Conversion postfix → prefix
- Linked list..
- Double linked list ..
- Multi linked list ..

2007-2008

تأليف :

صغير أحمد صغير الفصلي

جـــامعة صنـعـــاء
في الجمهورية اليمنيـة
كـلية العـــــــلوم
قسم الحاسوب والمعلومات

نعم هناك من لا يخطط وقد ينجح،،، ولكن الإستثناء ليس هو القاعدة ،،،

[1]-- Program of ....*stack*....

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define max 30
#define newline printf("\n");
#define space printf("\n\n\n\n\n\n\n\n");
struct stack
  {
    char ele[max];
    int top;
  }sag;
int i,n,m,r,t;
void initial (struct stack *);
void  read (struct stack *,char);
void delet(struct stack *);
main()
{
   clrscr();
   begin:
   printf("\n\nPress  --1-- to add symbols to stack \n-2-- to delete \n-3-- to display stack\n-4-- to EXIT .\n\n\t ");
   scanf("%d",&r);
   switch(r)
   {
    case 1:
         printf("\n\t    How many characters you want to enter\t");
         scanf("%d",&n);
         for(i=0;i<n;i++)
                  read(&sag,getche());
         printf("\n\n\t\t");
         getche();
         space
         goto begin;
    case 2:
         printf("\n\t\t\tHow many symbols you want to delete\t ");
         scanf("%d",&m);
         for(i=0;i<m;i++)
                  delet(&sag);
        printf("\n\n\t\t");
        getche();
        space
         goto begin;
   case 3:
        t=sag.top;
        printf("\n\t\t\t");
        for(i=0;i<=t;i++)
                  printf("%c",sag.ele[i]);
         printf("\n\n\n\n\t\t");
         getche();
         goto begin;
    case 4:
         exit(0);
   }
  return(0);
}
void initial(struct stack *ss)
{
      ss->top=-1;
}
void read(struct stack *ss,char z)
{
     if(ss->top<max)
          ss->ele[++ss->top]=z;
     else
          printf("\n\tstack is full\n\t");
}
void delet(struct stack *ss)
{

     if(ss->top<=-1)
           printf("\n\t stack empty\n");
     else
           ss->ele[ss->top--];
}
```


saghir55@yahoo.com صغير الفصلي

[2]-- Program of *....queue....*

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define space printf("\n\n\n\n\n\n\n\n\n\n\n\n");
#define max 30
struct queu
  {
    char ele[max];
    int rear,front;
  }sag;
int i,n=0,m,j,r;
void initial (struct queu *);
void read (struct queu *,char);
void delet(struct queu *);
char x,y;
main()
{       clrscr();
          begin:
          printf("-1-- to add symbols to Queue \n-2-- to delet \n-3-- to display\n-4-- to shift Queue to lift\n-5-- to EXIT .\n\t");
          scanf("%d",&r);
         switch(r)
         {
      case 1:
            printf("\n\t\tHow many characters you want to enter\t  ");
            scanf("%d",&n);
            for(i=0;i<n;i++)
               read(&sag,getche());
             getche();    space           goto begin;
     case 2:
           printf("\n\tHow many symbols you want to delete\t\t    ");
           scanf("%d",&m);
           for(i=0;i<=m;i++)
             delet(&sag);
           getche();         space            goto begin;
     case 3:
           for(i=sag.front;i<=sag.rear;i++)
            printf("%c",sag.ele[i]);
           getche();         space        goto begin;
     case 4:
          for(j=0;j<m;j++)
          {
                   for(i=sag.front;i<n-m;i++)
                   sag.ele[i-1]=sag.ele[i++];
          }
          printf("\n\tAfter shift the elements in queue to left \n\t\t\t  ");
          for(i=sag.front;i<=sag.rear;i++)
                  printf("%c",sag.ele[i]);
          getche();         space        goto begin;
    case 5:
           exit(0);
         }
         return(0);
}
void initial(struct queu *ss)
{
         ss->rear=-1;
         ss->front=0;
}
void read(struct queu *ss,char z)
{
         if(ss->rear<max)
               ss->ele[++ss->rear]=z;
         else
               puts("queu is full");
}
void delet(struct queu *ss)
{

         if(ss->rear< ss->front)
               printf("\n\t queu empty");
         else
               ss->ele[ss->front++];
}
```


saghir55@yahoo.com صغير الفصلي

[3]-- Program of **….conversion** *infix→postfix.…*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define max 30
struct stack
  {    char operato[max];
       int top ;
  }sag;
int prced(char topp,char symbol);
main()
{
          int i,r,k=0,n;
          char infix[max],postfix[max],z,f,l,x;
          clrscr();
          sag.top=-1;
                        /////////graphic
          printf("\n\n\t\t");
          for( i=0;i<44;i++)
                    printf("%c",196);
          printf("\n\t\t|-----What is length of expression----| \n");
          printf("\t\t");
          for( i=0;i<44;i++)
              printf("%c",196);
                              /////////////
          scanf("%d",&n);
          again:
          for(i=0;i<n;i++)
              infix[i]=getche();
                                                  /// to ensure that the input is infix   للتحقق أن المدخل بيني فعلاً
          f=infix[0]; l=infix[n-1];
          if(f=='$'||f==')'||f=='+'||f=='*'||f=='/'||f=='-'||l=='('||l=='!'||l=='+'||l=='-'||l=='*'||l=='%'||l=='/'||l=='$')
          {
                    printf("\tError.the expression which you enter is not infix..Please try again \n");
                    goto again;
          }
          for(r=0;r<n;r++)
          {
                    if((infix[r] >='a'&& infix[r]<='z')||(infix[r] >='A'&& infix[r]<='Z'))
                              postfix[k++]=infix[r];
                    else
                    {
                         while(sag.top != -1 && prced(sag.operato[sag.top],infix[r]))
                                   postfix[k++]=sag.operato[sag.top--];
                         if(infix[r] != ')')
                                   sag.operato[++sag.top]=infix[r];
                         else
                                   x=sag.operato[sag.top--];
                    }
          }
          while(sag.top>-1)
                   postfix[k++]=sag.operato[sag.top--];
          printf("  \n\n");
          for(i=0;i<k;i++)
                  printf("%c",postfix[i]);
          getche();
          return(0);
}
```


saghir55@yahoo.com صغير الفصلي

```
int prced(char topp,char symbol)        ////   function of prced .....دالة المقارنه
{
        int i,j,symb,toop;
        char operators[7][3]={ {'('},
                               {'|'},
                               {'&'},
                               {'=','>','<'},
                               {'+','-'},
                               {'*','/','%'},
                               {'!','$'}
                             };

         if(symbol=='(' || topp=='(' || (symbol==')'&&topp=='('))
           return(NULL);
         else
         {
                if(symbol==')')
                        return(1);
                else
                {
                        for(i=0;i<7;i++)
                          for(j=0;j<3;j++)
                            if(operators[i][j]==symbol)
                            {
                                symb=i;
                                break;
                            }
                        for(i=0;i<7;i++)
                          for(j=0;j<3;j++)
                            if(operators[i][j]==topp)
                            {
                                toop=i;
                                break;
                            }
                        if(toop>symb)
                             return(1);
                        else
                                return(NULL);
                }
         }
}
```

فكرتي لتكوين هذه الداله هو ترتيب المعاملات الحسابيه في مصفوفه ثنائية البعد.....بحيث يأخذ الأولويه بالترتيب وقمت بإدخال كل الرموز من نفس الدرجه في صف واحد .....أما عملية المقارنه فتتم إعتماداً على قيمه العداد (i) التي تتوقف عند قيمها (عند إيجاد المعامل)

saghir55@yahoo.com صغير الفصلي

[4]-- Program of **….conversion** *postfix → infix ….*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define max 30
struct stack
  {    char operand[max][max];
       int top ;
  }sag;
main()
{
  int i,k,n=0;
  char infix[max],postfix[max],z,op1[max],op2[max],sgn[max],exp[max],u,y;
  char f[2]={'('},l[2]={')'};
  clrscr();
  sag.top=-1;
  printf("\t\t What is length expression you want to enter \t");
  scanf("%d",&n);
  again:
  for(i=0;i<n;i++)
          postfix[i]=getche();
  u=postfix[0]; y=postfix[n-1];
  if(u=='$'||u==')'||u=='+'||u=='*'||u=='/'||u=='-'||u=='('||u=='!'||(y>='a'&&y<='z')||(y>='A'&&y<='Z'))
  {
             printf("\tError.the expression which you enter is not postfix..Please try again \n");
             goto again;
  }
  for(i=0;i<n;i++)
  {
           z=postfix[i];
           if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
           {
                    strcpy(sag.operad[++sag.top],empty);
                    sag.operand[sag.top][0]=z;
           }
           else
           {
                    strcpy(op2,sag.operand[sag.top--]);
                    strcpy(op1,sag.operand[sag.top--]);
                    sgn[0]=z;
                    strcpy(exp,f);
                    strcat(exp,op1);
                    strcat(exp,sgn);
                    strcat(exp,op2);
                    strcat(exp,l);
                    strcpy(sag.operand[++sag.top],exp);
           }
  }
  strcpy(infix,sag.operand[sag.top--]);
  printf("\n\t\t %s",infix);
  getche();
  return(0);
}
```


saghir55@yahoo.com صغير الفصلي

[5]-- Program of **….conversion** *infix→prefix….*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define max 30
struct stack
  {       char stackopen[max][max],stackoper[max];
          int top1,top2 ;
  }sa;
      char prefix[max],infix[max],z,ww,op1[max],op2[max],sgn[2],giv,exp[max],g,gg[max]={' '}, empty[2]={'  '};
void join(void);
main()
{
         int i,k,n=0;
         clrscr();
         sa.top1=-1;          sa.top2=-1;
         printf("\t\t What is length expression you want to enter \t");
         scanf("%d",&n);
          for(i=0;i<n;i++)
             infix[i]=getche();
         for(i=0;i<n;i++)
         {       z=infix[i];
                 if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
                  {           strcpy(sa.stackopen[++sa.top1],empty);
                              sa.stackopen[sa.top1][0]=z;
                   }
                 else
                 {       if(z==')')
                          {
                             while(sa.stackoper[sa.top2] !='(')
                                      join();
                             giv=sa.stackoper[sa.top2--];                    }
                          else
                          {      g=sa.stackoper[sa.top2];
                                 while((sa.top2 !=-1) && ((g=='/' || g=='*'|| g=='%'|| g=='$') && (z=='-' || z=='+')))
                                        join();
                                  sa.stackoper[++sa.top2]=z;
                          }
                  }
         }
         while (sa.top2 !=-1)
             join();
        printf("\n\n\n\t\t\t %s",sa.stackopen[sa.top1--]);
         getche();
         return(0);
}
void join()
{            strcpy(op2,sa.stackopen[sa.top1]);
             strcpy(sa.stackopen[sa.top1--],gg);
             strcpy(op1,sa.stackopen[sa.top1--]);
             ww=sa.stackoper[sa.top2--];
             sgn[0]=ww;
             strcpy(exp,sgn);
             strcat(exp,op1);
             strcat(exp,op2);
             strcpy(sa.stackopen[++sa.top1],exp);
  return;
}
```


saghir55@yahoo.com صغير الفصلي

[6]-- Program of **….conversion** *prefix → infix ….*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define max 30
struct pre
  {
     char  stack1[max],stack2[max][max];
     int top1,top2;
  }sag;
int i,j=0,n;
char prefix[max],infix[max],z,x,v,op1[max],op2[max],exp[max],sgn[2];
char qq[2]={'('},pp[2]={')'},empty[2]={' '};
main()
{            clrscr();
             sag.top1=-1;    sag.top2=-1;
             printf("\n\n\t \n\t\tconvertion prefix to infix \n\n"
                      " Length of the expession which you want to enter \t");
             scanf("%d",&n);
             for(i=0;i<n;i++)
                    prefix[i]=getche();
             for(i=0;i<n;i++)
                    sag.stack1[++sag.top1]=prefix[i];
             while(sag.top1 != -1)
             {       v=sag.stack1[sag.top1--];
                    if((v >='a'&& v<='z')||(v >='A'&& v<='Z'))
                    {
                                 strcpy(sag.stack2[++sag.top2],empty);
                                 sag.stack2[sag.top2][0]=v;
                    }
                    else
                    {
                                 strcpy(op1,sag.stack2[sag.top2--]);
                                 strcpy(op2,sag.stack2[sag.top2--]);
                                 sgn[0]=v;
                                 strcpy(exp,qq);
                                 strcat(exp,op1);
                                 strcat(exp,sgn);
                                 strcat(exp,op2);
                                 strcat(exp,pp);
                                 strcpy(sag.stack2[++sag.top2],exp);
                    }
             }
          strcpy(infix,sag.stack2[sag.top2--]);
          printf("\n\n%s",infix);
          getche();
          return(0);
}
```


saghir55@yahoo.com صغير الفصلي

[7]-- Program of **….conversion** *prefix → postfix ….*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define max 30
struct stack
  {    char operand[max][max],operr[max];
       int top,top2 ;
  }sag;
main()
{
        int i,k,n=0;
        char prefix[max],postfix[max],z,op1[max],op2[max],sgn[max],empty[2]={'  '};
        clrscr();
        sag.top=-1;
        sag.top2=-1;
        printf("\t\t What is longth expretion you want to enter \t");
        scanf("%d",&n);
        again:
        for(i=0;i<n;i++)
                prefix[i]=getche();
        for(i=0;i<n;i++)
                sag.operr[++sag.top2]=prefix[i];
        while(sag.top2!=-1)
        {       z=sag.operr[sag.top2--];
                if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
                {       strcpy(sag.operand[++sag.top],empty);
                        sag.operand[sag.top][0]=z;
                }

                else
                {       strcpy(op1,sag.operand[sag.top--]);
                        strcpy(op2,sag.operand[sag.top--]);
                        sgn[0]=z;
                        strcat(op1,op2);
                        strcat(op1,sgn);
                        strcpy(sag.operand[++sag.top],op1);
                }
         }
         strcpy(postfix,sag.operand[sag.top--]);
         printf("\n\t\t %s",postfix);
         getche();
         return(0);
}
```


saghir55@yahoo.com صغير الفصلي

[8]-- Program of **….conversion** *postfix → prefix….*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define max 30
struct stack
  {         char operand[max][max];
            int top ;
  }sag;
main()
{       int i,k,n=0;
        char infix[max],postfix[max],z,op1[max],op2[max],sgn[max],f,l,give[max], empty[2]={' '};
        clrscr();
        sag.top=-1;
        printf("\t\t length of expression you want to enter \t");
        scanf("%d",&n);
        again:
        for(i=0;i<n;i++)
                postfix[i]=getche();
        f=postfix[0]; l=postfix[n-1];
        if(f=='$'||f==')'||f=='+'||f=='*'||f=='/'||f=='-'||f=='('||f=='!'||(l>='a'&&l<='z')||(l>='A'&&l<='Z'))
        {         printf("\n\n\tError.the expression is not postfix..Please try again \n");
                  goto again;
        }
        for(i=0;i<n;i++)
        {
                 z=postfix[i];
                 if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
                 {          strcpy(sag.operand[++sag.top],empty);
                            sag.operand[sag.top][0]=z;
                  }
                 else
                 {       strcpy(op2,sag.operand[sag.top--]);
                         strcpy(op1,sag.operand[sag.top--]);
                         sgn[0]=z;
                         strcpy(give,sgn);
                         strcat(give,op1);
                         strcat(give,op2);
                         strcpy(sag.operand[++sag.top],give);
                 }
        }
        strcpy(infix,sag.operand[sag.top--]);
        printf("\n\t\t %s",infix);
        getche();
        return(0);
}
```


saghir55@yahoo.com صغير الفصلي

[9]-- Program of .... *Linked list ....*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct linklst
  {
    int name;
    struct linklst *link;
  };
typedef struct linklst node;
node  *f;
node *print(node *f);
node *getnode();
node * intitial(node *f);
node *preinsert(node *f);
node *postinsert(node *f);
node *ininsert(node *f);
node * initial(node *f);
node *predelete(node *f);
node *postdelete(node *f);
node *delet(node *f,int);
node * reorder(node *f);
node * reverse(node *f);
main()
{
        int c,n,i,r;
        clrscr();
        start:
        printf("\n--1-- to intitial  the array of nodes \n--2-- to insert to front \n--3-- to insert to end\n
        --4-- to insert at in order\n--5-- to delete from front\n--6-- to delete from end\n
        --7-- to delete by key\n--8-- to show the nodes\n--9-- to reorder  the nodes \n
        --10-- to reverse link lists\n--11--to EXIT .\n\n\t  ");
        scanf("%d",&c);
        switch(c)
        {
        case 1:
                f=initial(f);
                clrscr();
                goto start;
        case 2:
                n=0;
                printf("\n\t\tmany of linked list you want insert\t");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                     f=preinsert(f);
                 clrscr();
                goto start;
        case 3:
                printf("\n\t\tHow many linked list you want insert\t");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                   f=postinsert(f);
                clrscr();
                goto start;
        case 4:
                printf("\n\t\tHow many linked list you want insert\t");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                  f=ininsert(f);
                clrscr();
```


saghir55@yahoo.com صغير الفصلي

```
                goto start;
            case 5:
                printf("\n\n\t\tHow many nodes you want to delete from begining\t");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                    f=predelete(f);
                clrscr();
                goto start;
            case 6:
                printf("\n\n\t\tHow many nodes you want to delete from end\t");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                    f=postdelete(f);
                clrscr();
                goto start;
            case 7:
                printf("\n\n\t\tHow many nodes you want to delete \t");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                {
                        printf("\n\t\tenter value to delete");
                        scanf("%d",&r);
                       f=delet(f,r);
                }
                clrscr();
                goto start;
            case 8:
                f=print(f);
                getche();
                clrscr();
                goto start;
            case 9:
                f= reorder(f);
                clrscr();
                goto start;
            case 10:
                f=reverse(f);
                clrscr();
                goto start;
            case 11:
                exit(0);
            }
            return(0);
}
node *getnode()
{
             return((node*)malloc(sizeof(node)));
}
node * initial(node *f)
{
            f=getnode();
            printf("\n\n\t\t\tenter valeu   ");
            scanf("%d",&f->name);
            f->link=NULL;
            return f;
}
node *preinsert(node *f)
{
            node *u;
```


saghir55@yahoo.com صغير الفصلي

```
            u=getnode();
            printf("\n\n\t\t\t enter value  ");
            scanf("%d",&u->name);
            u->link=f;
            f=u;
          return(f);
}
node *postinsert(node *f)
{
            node *pp=f,*p;
            p=getnode();
            printf("\n\n\t\t\t enter value  ");
            scanf("%d",&p->name);
            while(pp->link)
                 pp=pp->link;
            pp->link=p;
            p->link=NULL;
            return(f);
}
node *ininsert(node *f)
{
            node *q,*p;
            p=getnode();
            printf("\n\n\t\t\t enter value  ");
            scanf("%d",&p->name);
            q=f;
            while(p->name>q->name && p->name > q->link->name)
               q=q->link;
            p->link=q->link;
            q->link=p;
            return(f);
}
node *print(node *f)
{
             node *p=f;
             while(p)
             {
                printf("\n\t\t%d",p->name);
                p=p->link;
             }
          return(f);
          }
node *predelete(node *f)
{
            f=f->link;
            return(f);
}
node *postdelete(node *f)
{
            node *d=f;
            while(d->link->link)
              d=d->link;
            d->link=NULL;
            return(f);
}
```


saghir55@yahoo.com صغير الفصلي

```
node *delet(node *f,int r)
{           node *d=f;
            if(r==f->name)
               f=f->link;
            else
            {       while((d->link->name != r) && ( d->link->link !=NULL))
                        d=d->link;
                    if(d->link->name==r)
                        d->link=d->link->link;
                    else
                        printf("\n\t\tThe number which you entered did not found");
            }
            return(f);
}
node *reverse(node *f)
{
            node *q=f,*p=NULL,*r;
            while(q)
            {
                    r=q->link;
                    q->link=p;
                    p=q;
                    f=q;
                    q=r;
            }
            return(f);
}
node * reorder(node *f)
{
             node *q,*p,*g,*a;
             int x=0;
             q=f;
             while(q->link)
             {  x++;
                p=q->link;
                g=q;
                while(p)
                {
                        if(q->name > p->name)
                        {       g->link=p->link;
                                p->link=q;
                                q=p;
                                p=q->link;
                                g=q;
                                a->link=q;
                        }
                         else
                         {      p=p->link;
                                g=g->link;
                         }
                }
                if(x==1)
                       f=q;
                a=q;
                q=q->link;
              }
            return(f);
}
```


saghir55@yahoo.com صغير الفصلي

[10]-- Program of .... *Double Linked list ....*

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct linklst
 {
        int val;
        struct linklst *left,*right;
 };
typedef struct linklst node;
node  *l,*u;
node *add(node *l);
node *add2r(node *l);
node *getnode(void);
node *initial(node *l);
node *delet(node *l,int key);
node *reorder(node *l);
void disply(node *l);
node *search(node *l,int key);
node *getpre(node *l,int key);
void main()
{
         int c,n,i,r,b;
         clrscr();
         start:
         printf("\n--1-- to intitial  the array of nodes \n--2-- to insert  \n--3-- to insert to right\n--4-- to
        delete\n--5-- to display\n--6-- to reorder\n--7-- to search\n--8-- to EXIT .\n\n\t  ");
         scanf("%d",&c);
         switch(c)
         {
              case 1:
                    l=initial(l);
                    clrscr();
                    goto start;
              case 2:
                    printf("\n\t\tno of nodes to enter ");
                    scanf("%d",&n);
                    for(i=0;i<n;i++)
                          l=add(l);
                    clrscr();
                    goto start;
              case 3:
                    l=add2r(l);
                    getche();
                    clrscr();
                    goto start;
              case 4:
                    printf("\n\t no of nodes to delete ");
                    scanf("%d",&n);
                    for(i=0;i<n;i++)
                    {
                          printf("\n\t enter value to delete");
                          scanf("%d",&b);
                          l=delet(l,b);
                    }
                    getche();
```


saghir55@yahoo.com صغير الفصلي

```
                clrscr();
                goto start;
            case 5:
                disply(l);
                getche();
                clrscr();
                goto start;
            case 6:
                l=reorder(l);
                getche();
                clrscr();
                goto start;
            case 7:
                printf("\n\t value to search ");
                scanf("%d",&b);
                u=search(l,b);
                printf(" \n\t\tthe value is %d ",u->val);
                getche();
                clrscr();
                goto start;
            case 8:
                exit(0);
        }
}
node *getnode()
{   return((node*)malloc(sizeof(node)));
}
node *initial(node *l)
{
        l=getnode();
        l->right=NULL;
        l->left=NULL;
        printf("\n\t\t enter the value ");
        scanf("%d",&l->val);
        return(l);
}
node *add(node *l)
{
        node *p,*s;
        p=getnode();
        printf("\n\t\tenter the value");
        scanf("%d",&p->val);
        s=getpre(l,p->val);
        p->left=s;
        p->right=s->right;
        s->right->left=p;
        s->right=p;
        return(l);
}
node *add2r(node *l)
{
        node *a;
        a=getnode();
        a->right=NULL;
        a->left=l;
        l->right=a;
        printf("\n\t\t enter value");
        scanf("%d",&a->val);
        return(l);
}
```


saghir55@yahoo.com صغير الفصلي

```
node *delet(node *l,int key)
{
        node *d,*p;
        p=search(l,key);
        p->right->left=p->left;
        p->left->right=p->right;
        return(l);
}
node *reorder(node *l)
{
        node *q=l,*p;
        int x=0;
        while(q->right)
        {
            x++;
            p=q->right;
            while(p)
            {
                    if(q->val>p->val)
                    {
                            p->left->right=p->right;
                            p->right->left=p->left;
                            p->right=q;
                            q->left->right=p;
                            p->left=q->left;
                            q->left=p;
                            q=p;
                            p=q->right;
                    }
                    else
                             p=p->right;
            }
            if(x==1)
                l=q;
            q=q->right;
         }
         return(l);
}
void disply(node *l)
{
        node *d=l;
        while(d)
        {
              printf("\n\t\t%d",d->val);
              d=d->right;
        }
}
node *search(node *l,int key)
{       node *f=l;
        while(f->val !=key &&f)
              f=f->right;
        return(f);
}
node *getpre(node *l,int key)
{
         node *y=l;
         while(y->val<key &&y->right->val<key &&y)
                 y=y->right;
          return(l);
}
```


saghir55@yahoo.com صغير الفصلي

[11]-- Program of .... *Multi Linked list ....*

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<alloc.h>
struct mnode
 {
        int row,col;
        int val;
        struct mnode *nextr,*nextc;
 };
typedef struct mnode node;
node* getnode();
node* initial(node*a,int r,int c);
node* insert(node*a);
node* delet(node *a,int r,int c);
node* disp(node *a);
node* getabov(node* a,int r,int c);
node* getlift(node* a,int r,int c);
int i,n=0;
void main()
{
          node* a;
          int rr,cc,m=0,mr,mc,j;
          clrscr();
          start:
           printf("\n\t\t\t1- initial\n\t\t\t2- insert \n\t\t\t3- delete \n\t\t\t4- dispay \n\t\t\t5- Exit\n");
           scanf("%d",&m);
           switch(m)
            {
                  case 1:
                        printf("\n\tenter size of matrix (row,col)\n");
                        scanf("%d%d",&mr,&mc);
                        a=initial(a,mr,mc);
                        goto start;
                  case 2:
                        printf("\n\t How many elements you want insert in the matrix\t");
                        scanf("%d",&n);
                        for(i=0;i<n;i++)
                        {
                              a=insert(a);
                        }
                        goto start;
                  case 3:
                        printf("\n\tHow many elements you want delete  ");
                        scanf("%d",&n);
                        for(i=0;i<n;i++)
                         {
                               printf("\n\t Enter location of element which you want delete\n");
                               scanf("%d%d",&rr,&cc);
                               a=delet(a,rr,cc);
                        }
                        goto start;
                  case 4:
                        a=disp(a);
                        goto start;
```


saghir55@yahoo.com صغير الفصلي

```
                case 5:
                        exit(0);
            }
}
node* getnode(void)
{
            return((node*)malloc(sizeof(node)));
}
node* initial(node *a,int rr,int cc)
{           node*s,*l;
            a=getnode();
            a->row=-1;
            a->col=-1;
            a->nextr=a;
            a->nextc=a;
            l=a;
            for(i=0;i<cc;i++)
            {         s=getnode();
                      s->row=-1;
                      s->col=i;
                      s->nextc=a;
                      s->nextr=s;
                      l->nextc=s;
                      l=s;
            }
            l=a;
            for(i=0;i<rr;i++)
            {        s=getnode();
                      s->row=i;
                      s->col=-1;
                      s->nextr=a;
                      s->nextc=s;
                      l->nextr=s;
                      l=s;
            }
            return(a);
}
node* insert(node*a)
{         node*left,*abov;
          int r,c;
          node *sa;
          printf("\n\n\t location of element which you will enter\n");
          scanf("%d%d",&r,&c);
          sa=getnode();
          sa->row=r;
          sa->col=c;
          printf("\n\t\t\tEnter value  ");
          scanf("%d",&sa->val);
          abov=getabov(a,r,c);
          left=getlift(a,r,c);
          sa->nextr=abov->nextr;
          abov->nextr=sa;
          sa->nextc=left->nextc;
          left->nextc=sa;
         return(a);
}
node* getabov(node *a,int r,int c)
{
            node *p=a->nextc;
            node *q;
```


saghir55@yahoo.com صغير الفصلي

```
        while(p!=a && p->col!=c)
            p=p->nextc;
        q=p;
        p=p->nextr;
        while(p->row !=-1 && p->row<r)
        {
            q=p;
            p=p->nextr;
        }
        return q;
}
node* getlift(node* a,int r,int c)
{
        node *p=a->nextr;
        node *q;
        while(p!=a && p->row!=r)
          p=p->nextr;
        q=p;
        p=p->nextc;
        while(p->col !=-1 && p->col<c)
        {
            q=p;
             p=p->nextc;
        }
        return q;
}
node* delet(node*a,int r,int c)
{
        node*abov,*left,*d;
        abov=getabov(a,r,c);
        left=getlift(a,r,c);
        d=abov->nextr;
        abov->nextr=d->nextr;
        left->nextc=d->nextc;
        free(d);
        return(a);
}
node* disp(node *a)
{
        int e,o;
        node *h,*print=a->nextr;
         while(print!=a)
        {
            h=print->nextc;
            while(h!=print)
              {
                  printf(" \n\t [%d][%d]  =%d",h->row,h->col,h->val);
                  h=h->nextc;
              }
              print=print->nextr;
        }
        return(a);
}
```


saghir55@yahoo.com صغير الفصلي

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#define max 30
struct stack
  {
          char ele[max];
          int  top;
  }sags;
struct queu
  {
          char ele[max];
          int rear,front;
  }sagq;
struct in2pos
  {       char operato[max];
          int top ;
  }sagip;
struct pos2in
  {       char operand[max][max];
          int top ;
  }sagpi;
struct in2pre
  {          char stackopen[max][max],stackoper[max];
             int top1,top2 ;
  }saipr;
struct pre2in
  {
           char  stack1[max],stack2[max][max];
           int top1,top2;
  }sagpri;
struct pr2ops
  {       char operand[max][max],operr[max];
          int top,top2 ;
  }sagprp;
struct pos2pre
  {          char operand[max][max];
             int top ;
  }sagppr;
struct linklst
  {
          int name;
          struct linklst *link;
  };
struct dlinklst
  {
          int val;
          struct dlinklst *left,*right;
  };
struct mnodem
  {
       int row,col;
```


saghir55@yahoo.com صغير الفصلي

```
        int val;
        struct mnodem *nextr,*nextc;
  };
typedef struct linklst node;
node *f;
typedef struct dlinklst noded;
noded  *l,*uu;
typedef struct mnodem nodem;
nodem* a;

int i,j,n,m,t,r,ch,k=0,c,b,rr,cc,mr,mc;
char infix[max],postfix[max],prefix[max],empty[2]={' '},gg[max]={'
'},sgn[max],op1[max],op2[max],ff[2]={'('},ll[2]={')'},give[max],exp[max];
char x,z,fa,la,ww,giv,g,u,y,v;
void initials(struct stack *ss)
                                                    //////////////////////  1- stack
{
     ss->top=-1;
}
void reads(struct stack *ss,char z)
{
    if(ss->top<max)
            ss->ele[++ss->top]=z;
    else
            printf("\n\tstack is full\n\t");
}
void delets(struct stack *ss)
{

    if(ss->top<=-1)
             printf("\n\t stack empty\n");
    else
             ss->ele[ss->top--];
}

void stackmain()
{
  clrscr();
  initials(&sags);
  begins:
  printf("\n\n==========STACK==========\n\n\n-1-- to add symbols to stack \n-2—to
  delete \n-3-- to display stack\n-4-- to EXIT .\n\n\t ");
  scanf("%d",&r);
  switch(r)
  {
   case 1:
           printf("\n\t    How many characters you want to enter\t");
           scanf("%d",&n);
           for(i=0;i<n;i++)
             reads(&sags,getche());
           printf("\n\n\t\t");
           getche();
           goto begins;
   case 2:
           printf("\n\t\t\tHow many symbols you want to delete\t ");
```


saghir55@yahoo.com صغير الفصلي

```
            scanf("%d",&m);
            for(i=0;i<m;i++)
              delets(&sags);
            printf("\n\n\t\t");
            goto begins;
   case 3:
            t=sags.top;
            printf("\n\t\t\t");
            for(i=0;i<=t;i++)
              printf("%c",sags.ele[i]);
            printf("\n\n\n\n\t\t");
            getche();
            goto begins;
   case 4:
            clrscr();
            return;
   }
  return;
}

                    ////////////////  2- Queue
void initialq(struct queu *ss)
{
          ss->rear=-1;
          ss->front=0;
}
void readq(struct queu *ss,char z)
{
          if(ss->rear<max)
            ss->ele[++ss->rear]=z;
          else
            puts("queu is full");
}
void deletq(struct queu *ss)
{

          if(ss->rear< ss->front)
               printf("\n\t queu empty");
          else
                ss->ele[ss->front++];
}
queuemain()
{       clrscr();
            initialq(&sagq);
           begin:
           printf("\n\n=========Queue==========\n\n\n-1-- to add symbols to Queue \n-2--
         to
           delet \n-3-- to display\n-4-- to shift Queue to lift\n-5-- to EXIT .\n\t");
           scanf("%d",&r);
         switch(r)
         {
         case 1:
                      printf("\n\t\tHow many characters you want to enter\t  ");
                      scanf("%d",&n);
                      for(i=0;i<n;i++)
```


saghir55@yahoo.com صغير الفصلي

```
                    readq(&sagq,getche());
                getche();          goto begin;
        case 2:
                printf("\n\tHow many symbols you want to delete\t\t    ");
                scanf("%d",&m);
                for(i=0;i<m;i++)
                  deletq(&sagq);
                 getche();              goto begin;
        case 3:
                for(i=sagq.front;i<=sagq.rear;i++)
                printf("%c",sagq.ele[i]);
                getche();             goto begin;
        case 4:
                for(j=0;j<m;j++)
                {
                      for(i=sagq.front;i<n-m;i++)
                        sagq.ele[i-1]=sagq.ele[i++];
                }
                printf("\n\tAfter shift the elements in queue to left \n\t\t\t  ");
                for(i=sagq.front;i<=sagq.rear;i++)
                        printf("%c",sagq.ele[i]);
                getche();             goto begin;
      case 5:
                clrscr();
                return(0);
      }
      return(0);
}

                                                        ////////////////////// 3- infix to postfix
int prced(char topp,char symbol)        ////   function of prced .....
{
int i,j,symb,toop;
char operators[7][3]={ {'('},
                       {'|'},
                       {'&'},
                       {'=','>','<'},
                       {'+','-'},
                       {'*','/','%'},
                       {'!','$'}
                     };

 if(symbol=='(' || topp=='(' || (symbol==')'&&topp=='('))
   return(NULL);
 else
 {
    if(symbol==')')
        return(1);
    else
    {
         for(i=0;i<7;i++)
           for(j=0;j<3;j++)
             if(operators[i][j]==symbol)
             {
               symb=i;
```


saghir55@yahoo.com صغير الفصلي

```
                break;
               }
            for(i=0;i<7;i++)
              for(j=0;j<3;j++)
               if(operators[i][j]==topp)
               {
                toop=i;
                break;
               }
            if(toop>symb)
                return(1);
            else
                                return(NULL);
      }
   }
}
                    /////
ipmain()
{
        clrscr();
        sagip.top=-1;
                 /////////graphic
        printf("\n\n\t\t");
        for( i=0;i<44;i++)
        printf("%c",196);
        printf("\n\t\t|-----What is length of expression----| \n");
        printf("\t\t");
        for( i=0;i<44;i++)
            printf("%c",196);
                        ////////////
        scanf("%d",&n);
        again:
        for(i=0;i<n;i++)
        infix[i]=getche();
                                                        /// to ensure that the input is infix
        fa=infix[0]; la=infix[n-1];
        if(fa=='$'||fa==')'||fa=='+'||fa=='*'||fa=='/'||fa=='-'||la=='('||la=='!'||la=='+'||la=='-
        '||la=='*'||la=='%'||la=='/'||la=='$')
        {
                printf("\tError.the expression which you enter is not infix..Please try again \n");
                goto again;
       }
        for(r=0;r<n;r++)
        {
               if((infix[r] >='a'&& infix[r]<='z')||(infix[r] >='A'&& infix[r]<='Z'))
                        postfix[k++]=infix[r];
               else
               {
                        while(sagip.top != -1 && prced(sagip.operato[sagip.top],infix[r]))
                              postfix[k++]=sagip.operato[sagip.top--];
                         if(infix[r] != ')')
                                  sagip.operato[++sagip.top]=infix[r];
                         else
                                  x=sagip.operato[sagip.top--];
                }
```


saghir55@yahoo.com صغير الفصلي

```
            }
        while(sagip.top>-1)
             postfix[k++]=sagip.operato[sagip.top--];
        printf("  \n\n");
        for(i=0;i<k;i++)
          printf("%c",postfix[i]);
        getche();
      //////////////00
        i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
      strcpy(infix,empty);
      strcpy(postfix,empty);
      strcpy(prefix,empty);
      strcpy(sgn,empty);
      strcpy(op1,empty);
      strcpy(op2,empty);
      strcpy(give,empty);
         ///////00
        clrscr();
        return(0);
}                    //////////////////////////

                                                                    ////////////   4- infix to prefix

void join()
{        strcpy(op2,saipr.stackopen[saipr.top1--]);
            strcpy(op1,saipr.stackopen[saipr.top1--]);
            ww=saipr.stackoper[saipr.top2--];
            sgn[0]=ww;
            strcpy(exp,sgn);
            strcat(exp,op1);
            strcat(exp,op2);
            strcpy(saipr.stackopen[++saipr.top1],exp);
  return;
}
          ////////////
iprmain()
{

  clrscr();
  saipr.top1=-1;          saipr.top2=-1;
  printf("\n convrsion infix to prefix\n\n\t\t What is length expression you want to enter \t");
  scanf("%d",&n);
   for(i=0;i<n;i++)
          infix[i]=getche();
  for(i=0;i<n;i++)
  {       z=infix[i];
    if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
     {      strcpy(saipr.stackopen[++saipr.top1],empty);
              saipr.stackopen[saipr.top1][0]=z;
          }
    else
    {       if(z==')')
             {
                        while(saipr.stackoper[saipr.top2] !='(')
                              join();
```


saghir55@yahoo.com صغير الفصلي

```
                    giv=saipr.stackoper[saipr.top2--];
            }
            else
            {     g=saipr.stackoper[saipr.top2];
                  while((saipr.top2 !=-1) && ((g=='/' || g=='*'|| g=='%'|| g=='$') && (z=='-' || z=='+')))
                       join();
                      saipr.stackoper[++saipr.top2]=z;
            }
       }
   }
   while (saipr.top2 !=-1)
           join();
  printf("\n\n\n\t\t\t %s",saipr.stackopen[saipr.top1--]);
   getche();
   clrscr();
//////////////00
   i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
strcpy(infix,empty);
strcpy(postfix,empty);
strcpy(prefix,empty);
strcpy(sgn,empty);
strcpy(op1,empty);
strcpy(op2,empty);
strcpy(give,empty);
       ///////00
   return(0);
}
                                                          ////////// 5- postfix to infix
pinmain()
{
   clrscr();
   sagpi.top=-1;
   printf("\n convrsion postfix to infix\n\n\t\t What is length expression you want to enter \t");
   scanf("%d",&n);
   again:
   for(i=0;i<n;i++)
          postfix[i]=getche();
   u=postfix[0]; y=postfix[n-1];
  if(u=='$'||u==')'||u=='+'||u=='*'||u=='/'||u=='-'||u=='('||u=='!'||(y>='a'&&y<='z')||(y>='A'&&y<='Z'))
   {
            printf("\tError.the expression which you enter is not postfix..Please try again \n");
            goto again;
   }
   for(i=0;i<n;i++)
   {
           z=postfix[i];
           if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
           {
                                strcpy(sagpi.operand[++sagpi.top],empty);
                    sagpi.operand[sagpi.top][0]=z;
           }
           else
               {
            strcpy(op2,sagpi.operand[sagpi.top--]);
            strcpy(op1,sagpi.operand[sagpi.top--]);
```


saghir55@yahoo.com صغير الفصلي

```
            sgn[0]=z;
            strcpy(exp,ff);
            strcat(exp,op1);
            strcat(exp,sgn);
            strcat(exp,op2);
            strcat(exp,ll);
            strcpy(sagpi.operand[++sagpi.top],exp);
            }
   }
   strcpy(infix,sagpi.operand[sagpi.top--]);
   printf("\n\t\t %s",infix);
   getche();
   //////////////00
   i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
strcpy(infix,empty);
strcpy(postfix,empty);
strcpy(prefix,empty);
strcpy(sgn,empty);
strcpy(op1,empty);
strcpy(op2,empty);
strcpy(give,empty);
        ///////00
   clrscr();
   return(0);
}
                                                          ////// 6- postfix to prefix
pprmain()
{
   clrscr();
   sagppr.top=-1;
   printf("\nconversion postfix to prefix\n\n\t\t length of expression you want to enter \t");
   scanf("%d",&n);
   again:
   for(i=0;i<n;i++)
          postfix[i]=getche();
   fa=postfix[0]; la=postfix[n-1];
   if(fa=='$'||fa==')'||fa=='+'||fa=='*'||fa=='/'||fa=='-
'||fa=='('||fa=='!'||(la>='a'&&la<='z')||(la>='A'&&la<='Z'))
   {          printf("\n\n\tError.the expression is not postfix..Please try again \n");
        goto again;
   }
   for(i=0;i<n;i++)
   {
               z=postfix[i];
     if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
               {          strcpy(sagppr.operand[++sagppr.top],empty);
                  sagppr.operand[sagppr.top][0]=z;
      }
     else
     {       strcpy(op2,sagppr.operand[sagppr.top--]);
           strcpy(op1,sagppr.operand[sagppr.top--]);
           sgn[0]=z;
           strcpy(give,sgn);
           strcat(give,op1);
           strcat(give,op2);
```


saghir55@yahoo.com صغير الفصلي

```
            strcpy(sagppr.operand[++sagppr.top],give);
      }
   }
   strcpy(infix,sagppr.operand[sagppr.top--]);
   printf("\n\t\t %s",infix);
   getche();
   clrscr();
//////////////00
   i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
strcpy(infix,empty);
strcpy(postfix,empty);
strcpy(prefix,empty);
strcpy(sgn,empty);
strcpy(op1,empty);
strcpy(op2,empty);
strcpy(give,empty);
        ///////00
   return(0);
}
                                                          ////// 7- prefix to infix
primain()
{             clrscr();
    sagpri.top1=-1;    sagpri.top2=-1;
    printf("\n\n\t \n\t\tconvertion prefix to infix \n\n"
             " Length of the expession which you want to enter \t");
    scanf("%d",&n);
    for(i=0;i<n;i++)
            prefix[i]=getche();
    for(i=0;i<n;i++)
            sagpri.stack1[++sagpri.top1]=prefix[i];
    while(sagpri.top1 != -1)
    {       v=sagpri.stack1[sagpri.top1--];
            if((v >='a'&& v<='z')||(v >='A'&& v<='Z'))
            {
                strcpy(sagpri.stack2[++sagpri.top2],empty);
                sagpri.stack2[sagpri.top2][0]=v;
            }
            else
            {
               strcpy(op1,sagpri.stack2[sagpri.top2--]);
               strcpy(op2,sagpri.stack2[sagpri.top2--]);
               sgn[0]=v;
               strcpy(exp,ff);
               strcat(exp,op1);
               strcat(exp,sgn);
               strcat(exp,op2);
               strcat(exp,ll);
               strcpy(sagpri.stack2[++sagpri.top2],exp);
            }
    }
strcpy(infix,sagpri.stack2[sagpri.top2--]);
printf("\n\n%s",infix);
getche();
clrscr();
//////////////00
```


saghir55@yahoo.com صغير الفصلي

```
   i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
strcpy(infix,empty);
strcpy(postfix,empty);
strcpy(prefix,empty);
strcpy(sgn,empty);
strcpy(op1,empty);
strcpy(op2,empty);
strcpy(give,empty);
        ///////0
return(0);
}

                                                          /////////// 8- prefix to postfix
void prpmain()
{
   clrscr();
   sagprp.top=-1;
   sagprp.top2=-1;
   printf("\n\t conversion prefix to postfix\n\n\t\t What is longth expretion you want to enter
\t");
   scanf("%d",&n);
   again:
   for(i=0;i<n;i++)
           prefix[i]=getche();
   for(i=0;i<n;i++)
           sagprp.operr[++sagprp.top2]=prefix[i];
   while(sagprp.top2!=-1)
   {           z=sagprp.operr[sagprp.top2--];
           if((z >='a'&& z<='z')||( z>='A'&& z<='Z'))
           {           strcpy(sagprp.operand[++sagprp.top],empty);
                       sagprp.operand[sagprp.top][0]=z;
           }

           else
           {   strcpy(op1,sagprp.operand[sagprp.top--]);
               strcpy(op2,sagprp.operand[sagprp.top--]);
               sgn[0]=z;
               strcat(op1,op2);
               strcat(op1,sgn);
               strcpy(sagprp.operand[++sagprp.top],op1);
           }
    }
    strcpy(postfix,sagprp.operand[sagprp.top--]);
    printf("\n\t\t %s",postfix);
    getche();
//////////////00
i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
strcpy(infix,empty);
strcpy(postfix,empty);
strcpy(prefix,empty);
strcpy(sgn,empty);
strcpy(op1,empty);
strcpy(op2,empty);
strcpy(give,empty);
           ///////00
```


saghir55@yahoo.com صغير الفصلي

```
    clrscr();

}
                                                        ////////////// 9- linked list
node *getnodel(void)
{
  return((node*)malloc(sizeof(node)));
}
node * initiall(node *f)
{
 f=getnodel();
 printf("\n\n\t\t\tenter valeu   ");
 scanf("%d",&f->name);
 f->link=NULL;
 return f;
}
node *preinsert(node *f)
{
 node *u;
 u=getnodel();
 printf("\n\n\t\t\t enter value  ");
 scanf("%d",&u->name);
 u->link=f;
 f=u;
return(f);
}
node *postinsert(node *f)
{
 node *pp=f,*p;
 p=getnodel();
 printf("\n\n\t\t\t enter value  ");
 scanf("%d",&p->name);
 while(pp->link)
      pp=pp->link;
 pp->link=p;
 p->link=NULL;
 return(f);
}
node *ininsert(node *f)
{
 node *q,*p;
 p=getnodel();
 printf("\n\n\t\t\t enter value  ");
 scanf("%d",&p->name);
 q=f;
 while(p->name>q->name && p->name > q->link->name)
    q=q->link;
 p->link=q->link;
 q->link=p;
 return(f);
}
node *print(node *f)
{
  node *p=f;
  while(p)
```


saghir55@yahoo.com صغير الفصلي

```
   {
      printf("\n\t\t%d",p->name);
      p=p->link;
   }
return(f);
}
node *predelete(node *f)
{
 f=f->link;
 return(f);
}
node *postdelete(node *f)
{
 node *d=f;
 while(d->link->link)
   d=d->link;
 d->link=NULL;
 return(f);
}
node *delet(node *f,int r)
{           node *d=f;
 if(r==f->name)
    f=f->link;
 else
 {      while((d->link->name != r) && ( d->link->link !=NULL))
              d=d->link;
   if(d->link->name==r)
              d->link=d->link->link;
   else
              printf("\n\t\tThe number which you entered did not found");
 }
 return(f);
}
node *reverse(node *f)
{
 node *q=f,*p=NULL,*r;
 while(q)
 {
   r=q->link;
   q->link=p;
   p=q;
   f=q;
   q=r;
 }
 return(f);
}
node * reorder(node *f)
{
  node *q,*p,*g,*a;
  int x=0;
  q=f;
  while(q->link)
  {  x++;
     p=q->link;
     g=q;
```


saghir55@yahoo.com صغير الفصلي

```
      while(p)
      {
            if(q->name > p->name)
            {       g->link=p->link;
                    p->link=q;
                    q=p;
                    p=q->link;
                    g=q;
                    a->link=q;
            }
             else
             {       p=p->link;
                    g=g->link;
             }
      }
      if(x==1)
            f=q;
      a=q;
      q=q->link;
    }
 return(f);
}
void linklistmain()
{

 clrscr();
 start:
 printf("\n\n=============linked list===========\n\n\n\n--1-- to intitial  the array of
nodes \n--2-- to insert to front \n--3-- to insert to end\n"
 "--4-- to insert at in order\n--5-- to delete from front\n--6-- to delete from end\n"
 "--7-- to delete by key\n--8-- to show the nodes\n--9-- to reorder  the nodes \n"
 "--10-- to reverse link lists\n--11--to EXIT .\n\n\t  ");
 scanf("%d",&c);
 switch(c)
 {
 case 1:
  f=initiall(f);
  clrscr();
  goto start;
 case 2:
  n=0;
  printf("\n\t\tmany of linked list you want insert\t");
  scanf("%d",&n);
  for(i=0;i<n;i++)
          f=preinsert(f);
   clrscr();
  goto start;
 case 3:
  printf("\n\t\tHow many linked list you want insert\t");
  scanf("%d",&n);
  for(i=0;i<n;i++)
        f=postinsert(f);
  clrscr();
  goto start;
 case 4:
```


saghir55@yahoo.com صغير الفصلي

```
   printf("\n\t\tHow many linked list you want insert\t");
   scanf("%d",&n);
   for(i=0;i<n;i++)
     f=ininsert(f);
   clrscr();
   goto start;
  case 5:
    printf("\n\n\t\tHow many nodes you want to delete from begining\t");
    scanf("%d",&n);
    for(i=0;i<n;i++)
            f=predelete(f);
    clrscr();
    goto start;
  case 6:
    printf("\n\n\t\tHow many nodes you want to delete from end\t");
    scanf("%d",&n);
    for(i=0;i<n;i++)
            f=postdelete(f);
    clrscr();
    goto start;
  case 7:
    printf("\n\n\t\tHow many nodes you want to delete \t");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
          printf("\n\t\tenter value to delete");
          scanf("%d",&r);
                    f=delet(f,r);
    }
    clrscr();
    goto start;
  case 8:
    f=print(f);
    getche();
    clrscr();
    goto start;
  case 9:
    f= reorder(f);
    clrscr();
    goto start;
  case 10:
    f=reverse(f);
    clrscr();
    goto start;
  case 11:
    i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
    return;
  }
}
                                                  /////////////// 10- double linked list
noded *getnoded()
{       return((noded*)malloc(sizeof(noded)));
}
noded *initiald(noded *l)
{
```


saghir55@yahoo.com صغير الفصلي

```
 l=getnoded();
 l->right=NULL;
 l->left=NULL;
 printf("\n\t\t enter the value ");
 scanf("%d",&l->val);
 return(l);
}
noded *getpre(noded *l,int key)
{
  noded *y=l;
  while(y->val<key &&y->right->val<key &&y)
          y=y->right;
    return(l);
}
noded *search(noded *l,int key)
{             noded *f=l;
 while(f->val !=key &&f)
           f=f->right;
 return(f);
}

noded *add(noded *l)
{
 noded *p,*s;
 p=getnoded();
 printf("\n\t\tenter the value");
 scanf("%d",&p->val);
 s=getpre(l,p->val);
 p->left=s;
 p->right=s->right;
 s->right->left=p;
 s->right=p;
 return(l);
}
noded *add2r(noded *l)
{
 noded *a;
 a=getnoded();
 a->right=NULL;
 a->left=l;
 l->right=a;
 printf("\n\t\t enter value");
 scanf("%d",&a->val);
 return(l);
}
noded *delet(noded *l,int key)
{
 noded *d,*p;
 p=search(l,key);
 p->right->left=p->left;
 p->left->right=p->right;
 return(l);
}
noded *reorder(noded *l)
{
```


saghir55@yahoo.com صغير الفصلي

```
 noded *q=l,*p;
 int x=0;
 while(q->right)
 {
  x++;
  p=q->right;
  while(p)
  {
        if(q->val>p->val)
        {
          p->left->right=p->right;
          p->right->left=p->left;
          p->right=q;
          q->left->right=p;
          p->left=q->left;
          q->left=p;
          q=p;
          p=q->right;
        }
        else
           p=p->right;
  }
  if(x==1)
        l=q;
  q=q->right;
  }
  return(l);
}
void disply(noded *l)
{
 noded *d=l;
 while(d)
 {
    printf("\n\t\t%d",d->val);
    d=d->right;
 }
}

void doubllinkmain()
{
 clrscr();
 start:
 printf("\n\n============double linked list=========\n\n\n--1-- to intitial  the array of
nodes \n--2-- to insert  \n--3-- to insert to right\n--4-- to delete\n--5-- to display\n--6-- to
reorder\n--7-- to search\n--8-- to EXIT .\n\n\t  ");
 scanf("%d",&c);
 switch(c)
 {
 case 1:
    l=initiald(l);
    clrscr();
        goto start;
 case 2:
    printf("\n\t\tno of nodes to enter ");
    scanf("%d",&n);
```


saghir55@yahoo.com صغير الفصلي

```
        for(i=0;i<n;i++)
              l=add(l);
        clrscr();
        goto start;
    case 3:
        l=add2r(l);
        getche();
        clrscr();
        goto start;
    case 4:
        printf("\n\t no of nodes to delete ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
              printf("\n\t enter value to delete");
              scanf("%d",&b);
              l=delet(l,b);
        }
        getche();
        clrscr();
        goto start;
    case 5:
        disply(l);
        getche();
        clrscr();
        goto start;
    case 6:
        l=reorder(l);
        getche();
        clrscr();
        goto start;
    case 7:
        printf("\n\t value to search ");
        scanf("%d",&b);
        uu=search(l,b);
        printf(" \n\t\tthe value is %d ",uu->val);
        getche();
        clrscr();
        goto start;
    case 8:
        i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
        return;
    }
}
```


saghir55@yahoo.com صغير الفصلي

```
nodem* getnodem(void)
{
 return((nodem*)malloc(sizeof(nodem)));
}
nodem* initialm(nodem *a,int rr,int cc)
{            nodem*s,*l;
 a=getnodem();
 a->row=-1;
 a->col=-1;
 a->nextr=a;
 a->nextc=a;
 l=a;
 for(i=0;i<cc;i++)
 {          s=getnodem();
  s->row=-1;
  s->col=i;
  s->nextc=a;
  s->nextr=s;
  l->nextc=s;
  l=s;
 }
 l=a;
 for(i=0;i<rr;i++)
 {         s=getnodem();
  s->row=i;
  s->col=-1;
  s->nextr=a;
  s->nextc=s;
  l->nextr=s;
  l=s;
 }
 return(a);
}
nodem* getabov(nodem *a,int r,int c)
{
 nodem *p=a->nextc;
 nodem *q;
 while(p!=a && p->col!=c)
       p=p->nextc;
 q=p;
 p=p->nextr;
 while(p->row !=-1 && p->row<r)
 {
   q=p;
   p=p->nextr;
 }
 return q;
}
nodem* getlift(nodem* a,int r,int c)
{
 nodem *p=a->nextr;
 nodem *q;
 while(p!=a && p->row!=r)
  p=p->nextr;
```


saghir55@yahoo.com صغير الفصلي

```
 q=p;
 p=p->nextc;
 while(p->col !=-1 && p->col<c)
 {
   q=p;
              p=p->nextc;
 }
 return q;
}
nodem* insert(nodem*a)
{         nodem*left,*abov;
  int r,c;
  nodem *sa;
  printf("\n\n\t location of element which you will enter\n");
  scanf("%d%d",&r,&c);
  sa=getnodem();
  sa->row=r;
  sa->col=c;
  printf("\n\t\t\tEnter value  ");
  scanf("%d",&sa->val);
  abov=getabov(a,r,c);
  left=getlift(a,r,c);
  sa->nextr=abov->nextr;
  abov->nextr=sa;
  sa->nextc=left->nextc;
  left->nextc=sa;
return(a);
}

nodem* deletm(nodem*a,int r,int c)
{
 nodem*abov,*left,*d;
 abov=getabov(a,r,c);
 left=getlift(a,r,c);
 d=abov->nextr;
 abov->nextr=d->nextr;
 left->nextc=d->nextc;
 free(d);
            return(a);
}
nodem* disp(nodem *a)
{
 int e,o;
 nodem *h,*print=a->nextr;
  while(print!=a)
 {
h=print->nextc;
while(h!=print)
   {
    printf(" \n\t [%d][%d]  =%d",h->row,h->col,h->val);
    h=h->nextc;
   }
  print=print->nextr;
 }
 return(a);
```


saghir55@yahoo.com صغير الفصلي

```
}

void multilinkmain()
{

 clrscr();
 start:
  printf("\n\t========multi linked list=======\n\n\n\t\t\t1-  initial\n\t\t\t2-  insert \n\t\t\t3-
delete \n\t\t\t4-  dispay \n\t\t\t5-  Exit\n");
  scanf("%d",&m);
  switch(m)
   {
   case 1:
    printf("\n\tenter size of matrix (row,col)\n");
    scanf("%d%d",&mr,&mc);
    a=initialm(a,mr,mc);
    goto start;
   case 2:
    printf("\n\t How many elements you want insert in the matrix\t");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
              a=insert(a);
    }
    goto start;
   case 3:
    printf("\n\tHow many elements you want delete  ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
     {
              printf("\n\t Enter location of element which you want delete\n");
          scanf("%d%d",&rr,&cc);
          a=deletm(a,rr,cc);
    }
    goto start;
   case 4:
    a=disp(a);
    goto start;
   case 5:
    i=0;j=0;n=0;m=0;t=0;r=0;ch=0;k=0;j=0;c=0;b=0;rr=0;cc=0;mr=0;mc=0;
    return;
   }
}

main()
{
 clrscr();
 start:
  printf("\n\n\t####### main list ########\n\n\n\t\t 1- stack\n\t\t 2- Queue \n\t\t"
          " 3- conversion infix   to postfix\n\t\t 4- conversion infix   to prefix"
          "\n\t\t 5- conveRsion postfix to infix\n\t\t 6- conversion postfix to prefix\n\t\t"
          " 7- conversion prefix  to infix\n\t\t 8- conversion prefix  to postfix\n\t\t 9- linked
list...\n\t\t"
       " 10- double linked list \n\t\t 11- multi linked list \n\n\t\tExit \n\t");
  scanf("%d",&ch);
```


saghir55@yahoo.com صغير الفصلي

```
  switch(ch)
  {
   case 1: stackmain();
           clrscr();
           goto start;
   case 2:
           queuemain();
           clrscr();
           goto start;
   case 3:
           ipmain();
           clrscr();
           goto start;
   case 4:
          iprmain();
          clrscr();
          goto start;
   case 5:
          pinmain();
          clrscr();
          goto start;
   case 6:
          pprmain();
          clrscr();
          goto start;
   case 7:
          primain();
          clrscr();
          goto start;
   case 8:
          prpmain();
          clrscr();
          goto start;
   case 9:
          linklistmain();
          clrscr();
          goto start;
   case 10:
          doubllinkmain();
          clrscr();
          goto start;
   case 11:
           multilinkmain();
           clrscr();
       goto start;
   case 12:
       exit(0);
  }
getche();
return(0);
}
```


saghir55@yahoo.com صغير الفصلي