BÀI TẬP THỰC HÀNH CẤU TRÚC DỮ LIỆU
(Đã được test trên Dev C++ 4.9.9.2)

1) Vận dụng các phép toán trên danh sách đặc để viết chương trình nhập vào một danh sách các số nguyên và hiển thị danh sách vừa nhập ra màn hình. Thêm 1 phần tử có nội dung x vào danh sách tại vị trí p (trong đó x và p được nhập từ bàn phím). Xóa phần tử đầu tiên có nội dung x (nhập từ bàn phím) ra khỏi danh sách.

```
/*DS_MANG.C*/
#include <conio.h>
#include <stdio.h>

#define MaxLength  100
typedef int ElementType;
typedef int Position;

typedef struct{
      ElementType Elements[MaxLength];
      Position  Last;
} List;

/*Khoi tao danh sach rong*/
void MakeNull_List(List &L){
      L.Last=0;
}

/*Kiem tra danh sach rong*/
int Empty_List(List L){
      return L.Last==0;
}

/*Vi tri phan tu dau tien*/
Position First(List L){
      return 1;
}

/*Gia tri phan tu o vi tri p*/
ElementType Retrieve(Position p, List L){
   return L.Elements[p-1];
}

/*Vi tri sau vi tri phan tu cuoi cung*/
Position EndList(List L){
  return L.Last+1;
}


/*Vi tri sau vi tri p*/
Position Next(Position  p, List L){
  return p+1;
}

/*Xen 1 phan tu vao danh sach*/
void Insert_List(ElementType x, Position  p,  List &L){
      if(L.Last==MaxLength)
            printf("Danh sach day");
      else  if((p<1)||(p>L.Last+1))
            printf("Vi tri  khong  hop le");
            else{
```

```
                Position q;
                for(q=(L.Last-1)+1;q>p-1; q--)
                     L.Elements[q]=L.Elements[q-1];
                L.Elements[p-1]=x;
                L.Last++;
           }
}

/*Xoa 1 phan tu ra khoi danh sach*/
void Delete_List(Position  p, List  &L){
      if((p<1)||(p>L.Last))
            printf("Vi tri khong hop le");
      else  if(Empty_List(L))
            printf("Danh sach rong!");
            else{
                Position  q;
                for(q=p-1;q<L.Last-1;q++)
                     L.Elements[q]=L.Elements[q+1];
                L.Last--;
           }
}

/*Dinh vi 1 phan tu trong danh sach*/
Position  Locate(ElementType x, List L){
      Position  p;
      int Found=0;
      p=First(L);
      while((p!=EndList(L))&&(Found==0))
         if(Retrieve(p,L)==x)Found=1;
         else p=Next(p,L);
      return  p;
}

/*Doc vao 1 danh sach*/
void Read_List(List &L){
   ElementType x;
   Position p=1;
   printf("\nEnter a list of interger numbers. -1 to stop.\n");
   do{
      printf("x=");scanf("%d",&x);
      Insert_List(x,p,L);
      p++;
   }while(x!=-1);

}

/*In danh sach ra man hinh*/
void Print_List(List L){
   Position p;
   for(p=1;p!=EndList(L);p++){
      printf("%5d",Retrieve(p,L));
   }
   printf("\n");
}

main(){
      List L;
      ElementType  x;
      Position p;
```

```
      MakeNull_List(L);
      Read_List(L);
      printf("Danh sach vua nhap:");
      Print_List(L);
      printf("Phan  tu  can them:"); scanf("%d",&x);
      printf("Vi tri can them:"); scanf("%d",&p);
      Insert_List(x,p,L);
      printf("Danh sach  sau  khi them  phan tu la:");
      Print_List(L);
      printf("Noi  dung  phan  tu  can xoa:");scanf("%d",&x);
      p=Locate(x,L);
      if(p!=EndList(L)) Delete_List(p,L);
      printf("Danh sach  sau  khi xoa  %d la:",x);
      Print_List(L);
      getch();
}
```

Chú ý:

Dùng stdio.h cho các hàm vào ra chuẩn: scanf, printf, getc, putc, getchar, putchar, gets, puts, fflush.

Dùng conio.h cho các hàm liên quan đến điều khiển màn hình như: clrscr và getch.

2) Vận dụng các phép toán trên danh sách liên kết để viết chương trình nhập vào một danh sách các số nguyên. Sau đó thực hiện các công việc sau:

- **Hiển thị danh sách vừa nhập ra màn hình.**
- **Thêm 1 phần tử có nội dung x vào danh sách tại ví trí n (trong đó x và n được nhập từ bàn phím).**
- **Xóa phần tử đầu tiên có nội dung x (nhập từ bàn phím) ra khỏi danh sách.**
- **Sắp xếp danh sách theo thứ tự tăng dần.**

/*DSCONTRO.C*/

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
typedef int ElementType;
typedef struct Node{
      ElementType Element;
      struct Node *Next;
};
typedef struct Node* Position;
typedef Position List;

/*Tao danh sach rong*/
void MakeNull_List(List &Header){
      Header=(Node*)malloc(sizeof(Node));
      Header->Next=NULL;
}

/*Kiem tra danh sach rong*/
int Empty_List(List L){
      return (L->Next==NULL);
}
/*Xen mot phan tu vao danh sach*/
void Insert_List(ElementType X, Position P, List &L){
      Position T;
      T=(Node*)malloc(sizeof(Node));
      T->Element=X;
      T->Next=P->Next;
      P->Next=T;
}
```

```
/*Xoa 1 phan tu khoi danh sach*/
void Delete_List(Position P, List &L){
      Position T;
      if(P->Next!=NULL){
            T=P->Next;
            P->Next=T->Next;
            free(T);
      }
}
/*Dinh vi mot phan tu trong danh sach*/
Position Locate(ElementType X, List L){
      Position P;
      int Found=0;
      P=L;
      while((P->Next!=NULL)&&(Found==0))
            if(P->Next->Element==X)Found=1;
            else P=P->Next;
      return P;
}

/*Tra ve gia tri cua phan tu o vi tri P*/
ElementType Retrieve(Position P, List L){
      if(P->Next!=NULL)
          return P->Next->Element;
}

/*Xac dinh vi tri phan tu dau*/
Position First(List L){
      return L;
}

/*Xac dinh vi tri phan tu sau phan tu cuoi*/
Position EndList(List L){
      Position P;
      P=First(L);
      while(P->Next!=NULL)P=P->Next;
      return P;
}


Position Position_n(int n, List L){
      int i=1;
      Position P;
      P=L;
      while(P->Next!=NULL && i<n){
            P=P->Next;
            i=i+1;
      }
      return P;
}

/*Sap xep tang dan*/
void Sort(List &L){
      Position P,Q;
      ElementType temp;
      P=L->Next;
      while(P!=NULL){
            Q=P->Next;
```

```
            while(Q!=NULL){
                  if(P->Element > Q->Element){
                      temp=P->Element;
                      P->Element=Q->Element;
                      Q->Element=temp;
                  }
                  Q=Q->Next;
            }
            P=P->Next;
      }
}

/*Nhap danh sach*/
void  Read_List(List &L){
      ElementType x;
      printf("Nhap vao 1 danh sach cac so nguyen, -1 de dung\n");
      MakeNull_List(L);
      do{
         printf("x=");scanf("%d",&x);
         Insert_List(x,EndList(L),L);
      }while(x!=-1);
}

/*In danh sach ra man hinh*/
void Print_List(List L){
      Position P;
      P=L->Next;
      while(P!=NULL){
        printf("%5d",P->Element);
        P=P->Next;
      }
}

main(){
      ElementType x;
      int n;
      List L;
      Position P;

      Read_List(L);
      Print_List(L);
      printf("\nNhap vao 1 phan tu de them vao ds,  x=");
      scanf("%d",&x);
      printf("Vi tri can them, n=");scanf("%d",&n);
      P=Position_n(n,L);
      Insert_List(x,P,L);
      printf("\nDanh sach sau khi them %d la:\n",x);
      Print_List(L);
      printf("\nNhap vao 1 phan tu de xoa,  x=");scanf("%d",&x);
      P=Locate(x,L);
      if(P->Next!=NULL) Delete_List(P,L);
      printf("\nDanh sach sau khi xoa %d la:\n",x);
      Print_List(L);

      Sort(L);
      printf("\nDanh sach sau khi duoc sap xep:\n",x);
      Print_List(L);
      getch();
}
```

3) Viết chương trình nhập vào 1 số nguyên n. Chuyển số nguyên n sang số nhị phân (có sử dụng các phép toán của ngăn xếp).

/*DOISO_NP.C*/

```
#include <stdio.h>
#include <conio.h>
#define Maxlength 100
typedef int ElementType;
typedef struct{
    ElementType Elements[Maxlength];
    int Top_idx;
}Stack;

/*Khoi tao 1 ngan xep rong*/
void MakeNull_Stack(Stack &S){
    S.Top_idx=Maxlength;
}

/*Kiem tra ngan xep co rong khong*/
int Empty_Stack(Stack S){
    return (S.Top_idx==Maxlength);
}

/*Kiem tra ngan xep co day khong*/
int Full_Stack(Stack S){
    return (S.Top_idx==0);
}
/*Tra lai noi dung phan tu tren dinh*/
ElementType Top(Stack S){
    if(!Empty_Stack(S))
          return S.Elements[S.Top_idx];
    else printf("Error! Stack is empty");
}
/*Xoa 1 phan tu khoi ngan xep*/
void Pop(Stack &S){
    if(!Empty_Stack(S))
          S.Top_idx=S.Top_idx+1;
    else printf("Error! Stack is empty");
}

/*Day 1 phan tu vao ngan xep*/
void Push(ElementType x, Stack &S){
    if(Full_Stack(S))
          printf("Error!Stack is full");
    else{
          S.Top_idx=S.Top_idx-1;
          S.Elements[S.Top_idx]=x;
    }

}

/*Tim va in ra so nhi phan tuong ung*/
void print_binary(int n){
    Stack S;
    MakeNull_Stack(S);
    while(n!=0){
           Push(n%2,S);
           n=n/2;
    };
```

```
      printf("So nhi phan tuong ung la:");
      while(!Empty_Stack(S)){
            printf("%d",Top(S));
            Pop(S);
      }
  }
main(){
      int n;
      printf("Nhap vao 1 so nguyen:");scanf("%d",&n);
      print_binary(n);
      getch();
}
```

4) Viết chương trình nhập vào một ngăn xếp chứa các số nguyên. Sau đó sử dụng một hàng đợi để đảo ngược thứ tự của các phần tử trong ngăn xếp.

/*DAOSTACK.CPP*/

```
#include <stdio.h>
#include <conio.h>
#define Maxlength 100

/*Cai dat ngan xep*/
typedef int ElementType;
typedef struct{
      ElementType Elements[Maxlength];
      int Top_idx;
}Stack;

/*Khoi tao 1 ngan xep rong*/
void MakeNull_Stack(Stack &S){
      S.Top_idx=Maxlength;
}

/*Kiem tra xem 1 ngan xep co rong khong*/
int Empty_Stack(Stack S){
      return (S.Top_idx==Maxlength);
}



/*Kiem tra xem 1 ngan xep co day khong*/
int Full_Stack(Stack S){
      return (S.Top_idx==0);
}

/*Ham tra ve noi dung cua phan tu tren dinh*/
ElementType Top(Stack S){
      if(!Empty_Stack(S))
            return S.Elements[S.Top_idx];
      else printf("Error! Stack is empty");
}

/*Xoa 1 phan tu khoi ngan xep*/
void Pop(Stack &S){
      if(!Empty_Stack(S))
            S.Top_idx=S.Top_idx+1;
      else printf("Error! Stack is empty");
}
```

```
/*Day 1 phan tu len ngan xep*/
void Push(ElementType x, Stack &S){
      if(Full_Stack(S))
            printf("Error!Stack is full");
      else{
            S.Top_idx=S.Top_idx-1;
            S.Elements[S.Top_idx]=x;
      }

}


/*Cai dat hang doi*/
typedef struct{
      ElementType Elements[Maxlength];
      int Front, Rear;
}Queue;
/*Khoi tao 1 hang doi rong*/
void MakeNull_Queue(Queue &Q){
      Q.Front=-1;
      Q.Rear=-1;
}

/*Kiem tra xem 1 hang doi co rong khong*/
int Empty_Queue(Queue Q){
      return Q.Front==-1;
}
/*Kiem tra xem 1 hang doi co day khong*/
int Full_Queue(Queue Q){
      return (Q.Rear-Q.Front+1)%Maxlength==0;
}
/*Xoa bo 1 phan tu khoi hang doi*/
void DeQueue(Queue &Q){
      if(!Empty_Queue(Q)){
            if(Q.Front==Q.Rear)MakeNull_Queue(Q);
            else Q.Front=(Q.Front+1)%Maxlength;
      }
      else printf("Error! Queue is empty");
}


/*Them 1 phan tu vao hang*/
void EnQueue(ElementType x, Queue &Q){
      if(!Full_Queue(Q)){
            if(Empty_Queue(Q))Q.Front=0;
            Q.Rear=(Q.Rear+1)%Maxlength;
            Q.Elements[Q.Rear]=x;
      }
      else printf("Error! Queue is full");
}

/*Tra ve phan tu dau hang*/
ElementType Front(Queue Q){
     if(Empty_Queue(Q))
       printf("Hang rong!");
     else
       return Q.Elements[Q.Front];
}
```

```
main(){
      Stack S;
      Queue Q;
      ElementType x;

      printf("Nhap vao 1 danh sach so nguyen, -1 de dung:\n");
      /*Tao 1 ngan xep*/
      MakeNull_Stack(S);
      do{
         printf("x=");scanf("%d",&x);
         Push(x,S);
      }while(x!=-1);

      /*Tao 1 hang doi*/
      MakeNull_Queue(Q);
      while(!Empty_Stack(S)){
          EnQueue(Top(S), Q);
          Pop(S);
      }


      /*Di chuyen cac phan tu tu hang doi vao ngan xep*/
      while(!Empty_Queue(Q)){
         Push(Front(Q),S);
         DeQueue(Q);
      }

      /*Hien thi ngan xep*/
      printf("\nNgan xep sau khi dao nguoc (duoc lay ra tu dinh -> day):");
      while(!Empty_Stack(S)){
          printf("%d ",Top(S));
          Pop(S);
      }

      getch();
}
```

5) Viết chương trình xây dựng 1cây tổng quát để lưu trữ các ký tự. Sau đó thực hiện các công việc sau:

- **Nhập vào số nút của cây.**
- **Ứng với mỗi nút thì phải nhập nhãn và cha của nó.**
- **Hiển thị danh sách duyệt cây theo các thứ tự tiền tự, trung tự và hậu tự.**

```
/*CAYTQUAT.C*/
#include <stdio.h>
#include <conio.h>
#define Maxlength 100
#define NIL -1
typedef char DataType;
typedef int Node;
typedef struct{
      DataType Data[Maxlength];
      Node Parent[Maxlength];
      int MaxNode;
}Tree;
Tree T;
/*Khoi tao 1 cay rong*/
void MakeNull_Tree(Tree &T){
      T.MaxNode=0;
}
/*Kiem tra xem 1 cay co rong khong*/
int EmptyTree(Tree T){
      return T.MaxNode==0;
}
/*Xac dinh cha cua 1 nut*/
Node Parent(Node n, Tree T){
      if(EmptyTree(T)||(n>T.MaxNode-1))
            return NIL;
      else return T.Parent[n];
}
/*Tra ve nhan cua 1 nut*/
DataType Label_Node(Node n,Tree T){
      if(!EmptyTree(T)&&(n<=T.MaxNode-1))
            return T.Data[n];
      else return '*';
}

/*Xac dinh nut goc cua cay*/
Node Root(Tree T){
      if(!EmptyTree(T))return 0;
      else return NIL;
}
/*Xac dinh nut con trai nhat cua 1 nut*/
Node LeftMostChild(Node n, Tree T){
      Node i;
      int found;
      if(n<0) return NIL;
      i=n+1;
      found=0;
      while((i<=T.MaxNode-1)&&!found)
            if(T.Parent[i]==n)found=1;
            else i=i+1;
      if(found)return i;
      else return NIL;
}
```

```
/*Xac dinh nut anh em ruot phai cua 1 nut*/
Node Rightsibling(Node n, Tree T){
      Node i, parent;
      int found;
      if(n<0)return NIL;
      parent=T.Parent[n];
      i=n+1;
      found=0;
      while((i<=T.MaxNode-1)&&!found)
            if(T.Parent[i]==parent) found=1;
            else i=i+1;
      if(found)return i;
      else return NIL;
}
/*Duyet tien tu*/
void preorder(Node n, Tree T){
      Node i;
      printf("%5c",Label_Node(n,T));
      i=LeftMostChild(n,T);
      while(i!=NIL){
            preorder(i,T);
            i=Rightsibling(i,T);
      }
}
/*Duyet trung tu*/
void inorder(Node n, Tree T){
      Node i;
      i=LeftMostChild(n,T);
      if(i!=NIL)inorder(i,T);
      printf("%5c",Label_Node(n,T));
      i=Rightsibling(i,T);
      while(i!=NIL){
            inorder(i,T);
            i=Rightsibling(i,T);
      }
}
/*Duyet hau tu*/
void postorder(Node n, Tree T){
      Node i;
      i=LeftMostChild(n,T);
      while(i!=NIL){
            postorder(i,T);
            i=Rightsibling(i,T);
      }
      printf("%5c",Label_Node(n,T));
}

/*Nhap vao 1 cay*/
void ReadTree(Tree &T){
   int i;
   MakeNull_Tree(T);
   do{
      printf("Cay co bao nhieu nut? ");
      scanf("%d",&T.MaxNode);
   }while((T.MaxNode<1)||(T.MaxNode>Maxlength));

   printf("Nhap nhan (gia tri) cho nut goc (nut 0):");
   fflush(stdin);
   scanf("%c",&T.Data[0]);
```

```
   T.Parent[0]=NIL;

   printf("Nhap cac nut con lai\n");
   for(i=1;i<=T.MaxNode-1;i++){
      printf(" +Cha cua nut %d la: ",i);
      scanf("%d",&T.Parent[i]);
      printf("  Nhan (gia tri) cua nut %d la: ",i);
      fflush(stdin);
      scanf("%c",&T.Data[i]);
   }
}
main(){
      printf("Nhap du lieu cho cay tong quat, chua cac ky tu\n");
      ReadTree(T);
      printf("\nDuyet tien tu:");
      preorder(Root(T),T);
      printf("\nDuyet trung tu:");
      inorder(Root(T),T);
      printf("\nDuyet hau tu::");
      postorder(Root(T),T);
      getch();
}
```

6) Viết chương trình nhập vào 1 dãy các số nguyên để lưu vào 1 cây tìm kiếm nhị phân. Sau đó thực hiện các công việc sau:

- **Hiển thị danh sách duyệt cây theo các thứ tự tiền tự, trung tự và hậu tự.**
- **Tính chiều cao của cây.**
- **Đếm số nút của cây.**

```
/*CAYTKNP.C*/
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
typedef int KeyType;
typedef struct Node{
      KeyType Key;
      struct Node *left,*right;

};
typedef struct Node* Tree;

/*Khoi tao cay rong*/
void MakeNullTree(Tree &T){
      T=NULL;
}

/*Kiem tra cay rong*/
int EmptyTree(Tree T){
      return T==NULL;
}

/*Xac dinh con trai cua 1 nut*/
Tree LeftChild(Tree n){
      if(n!=NULL)return n->left;
      else return NULL;
}
```

```
/*Xac dinh con phai cua 1 nut*/
Tree RightChild(Tree n){
      if(n!=NULL)return n->right;
      else return NULL;
}
/*Kiem tra 1 nut co phai la nut la*/
int IsLeaf(Tree n){
      if(n!=NULL)
            return (LeftChild(n)==NULL)&&(RightChild(n)==NULL);
      else return 0;
}
/*Dem so nut cua cay*/
int nb_nodes(Tree T){
      if(EmptyTree(T))return 0;
     else return
             1+nb_nodes(LeftChild(T))+nb_nodes(RightChild(T));
}


/*Duyet tien tu*/
void PreOrder(Tree T){
      printf("%5d",T->Key);
      if(LeftChild(T)!=NULL) PreOrder(LeftChild(T));
      if(RightChild(T)!=NULL) PreOrder(RightChild(T));
}

/*Duyet trung tu*/
void InOrder(Tree T){
      if(LeftChild(T)!=NULL)InOrder(LeftChild(T));
      printf("%5d",T->Key);
      if(RightChild(T)!=NULL)InOrder(RightChild(T));
}

/*Duyet hau tu*/
void PosOrder(Tree T){
      if(LeftChild(T)!=NULL)PosOrder(LeftChild(T));
      if(RightChild(T)!=NULL)PosOrder(RightChild(T));
      printf("%5d",T->Key);
}

/*Chen  1 nut vao cay*/
Tree Search(KeyType x, Tree Root){
      if(Root==NULL) return NULL;
      else if(Root->Key==x)
            return Root;
      else if(Root->Key<x)
            return Search(x,Root->right);
      else
            return Search(x,Root->left);
}

/*Chen 1 nut vao cay*/
void InsertNode(KeyType x, Tree &Root){
      if(Root==NULL){
            Root=(Node*)malloc(sizeof(Node));
            Root->Key=x;
            Root->left=NULL;
            Root->right=NULL;
      }
```

```
      else if(x<Root->Key) InsertNode(x,Root->left);
      else if(x>Root->Key) InsertNode(x,Root->right);
}


/*Xoa nut co gia tri nho nhat (trai nhat) khoi cay TKNP
  cay nay co goc la root*/
KeyType DeleteMin(Tree &Root){
      KeyType k;
      if(Root->left==NULL){
            k=Root->Key;
            Root=Root->right;
            return k;
      }else return DeleteMin(Root->left);
}

/*Xoa 1 nut khoi cay*/
void DeleteNode(KeyType x, Tree &Root){
      if(Root!=NULL)
            if(x<Root->Key)DeleteNode(x,Root->left);
            else if(x>Root->Key) DeleteNode(x,Root->right);
            else if((Root->left==NULL)&&(Root->right==NULL))
                  Root=NULL;
            else if(Root->left==NULL) Root=Root->right;
            else Root->Key=DeleteMin(Root->right);
}

/*Nhap vao 1 cay*/
void ReadTree(Tree &T){
      int x;
      printf("Nhap vao cac so nguyen cho cay TKNP, -1 de dung:\n");
      MakeNullTree(T);
      do{
         printf("x="); scanf("%d",&x);
         if(x!=-1)InsertNode(x,T);
      }while(x!=-1);
}

/*Tim  max*/
int max(int a, int b){
  if(a>b)return a;
  else return b;
}

/*Chieu cao cua cay*/
int tree_height(Tree T){
      /*neu cay rong hay cay chi co 1 nut thi chieu cao=0*/
      if((T==NULL)||((LeftChild(T)==NULL) && (RightChild(T)==NULL)))return 0;
      else return 1+max(tree_height(LeftChild(T)),
tree_height(RightChild(T)));
}

int sonut1con(Tree T){
    if(T==NULL) return 0;
    else if (T->left==NULL && T->right!=NULL) return 1+sonut1con(T->right);
    else if (T->left!=NULL && T->right==NULL) return 1+sonut1con(T->left);
    else return sonut1con(T->left)+sonut1con(T->right);
    }
```

```
int main(){
      Tree T;
      ReadTree(T);
      printf("So nut cua cay la:%d",nb_nodes(T));
      printf("\nChieu cao cua cay la:%d",tree_height(T));
      printf("\nTien tu:");PreOrder(T);
      printf("\nTrung tu:");InOrder(T);
      printf("\nHau tu:");PosOrder(T);
      getch();
      return 0;
}
```

7) Viết chương trình nhập vào 2 tập hợp A và B để chứa các số nguyên có giá trị trong đoạn [0..99] theo dạng vectơ bit. Sau đó thực hiện các công việc sau:

- **Hiển thị 2 tập A và B.**
- **Hiển thị các tập giao, hợp và hiệu của A và B.**

/*TAP_HOP.C*/

```
#include <stdio.h>
#include <conio.h>
#define n 100      /*pham vi cua tap hop tu 0..99*/
typedef int SET[n];
/*Khoi tao tap hop rong*/
void makenullset(SET s){
      int i;
      for(i=0;i<=n-1;i++)
            s[i]=0;
}
/*Chen 1 phan tu vao  tap  hop*/
void insertset(int x, SET s){
     if((x>=0)&&(x<=n-1))
       s[x]=1;
}
/*Hop cua 2 tap hop*/
void set_union(SET a,SET b, SET c){
      int i;
      for(i=0;i<=n-1;i++)
            c[i]=a[i]||b[i];
}
/*Giao cua 2 tap hop*/
void set_intersection(SET a, SET b, SET c){
      int i;
      for(i=0;i<=n-1;i++)
            c[i]=a[i]&&b[i];
}
/*Hieu a\b*/
void set_differrence(SET a, SET b, SET c){
      int i;
      for(i=0;i<=n-1;i++)
            c[i]=a[i]&&(!b[i]);
}
/*Hien thi tap  hop*/
void print_set(SET s){
      int i;
      for(i=0;i<=n-1;i++)
            if(s[i]==1)printf("%5d",i);
      putchar('\n');
}
```

```
/* Nhap  vao 1 tap cac so nguyen */
void read_set(SET s){
      int x;
      printf("\nNhap vao cac so nguyen tu [0-99], -1 de dung.\n");
      makenullset(s);
      do{
        printf("x=");scanf("%d",&x);
        insertset(x,s);
      }while(x!=-1);
}


int main(){
      SET a,b,c,d,e;
      printf("SET A:");
      read_set(a);
      print_set(a);

      printf("SET B:");
      read_set(b);
      print_set(b);

      set_union(a,b,c);
      set_intersection(a,b,d);
      set_differrence(a,b,e);

      printf("A union B:");print_set(c);
      printf("A intersect B");print_set(d);
      printf("A differ B");print_set(e);
      getch();
      return 0;
}
```

8) Viết chương trình cài đặt một tự điển dùng bảng băm đóng để lưu trữ các từ tiếng Anh, như: "hello", "go", "study", v.v. B=100. Hàm băm *h(x)=Tổng giá trị ASCII của các ký tự trong từ %B*. Sau đó thực hiện các công việc sau:

- **Hiển thị tự điển.**
- **Nhập vào 1 từ. Tìm kiếm xem nó có trong tự điển hay không.**
- **Nhập vào 1 từ. Xóa nó khỏi tự điển. Hiển thị lại tự điển.**

/*BAM_DONG.C*/

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define B 100
#define empty "++++++++++"
#define deleted "**********"

typedef char* ElementType;
typedef ElementType Dictionary[B];
typedef int Position;

/*hash function*/
int h(ElementType x){
      int i,sum=0;
      for(i=0;i<strlen(x);i++)sum=sum+x[i];;
      return sum%B;
}
```

```
/*create a empty dictionary*/
void makenulldict(Dictionary D){
      int i;
      for(i=0;i<B;i++){
           D[i]=strdup(empty);
           }
}

/*identify if a member is in the dictionary*/
int locate(ElementType x, Dictionary D){
      int initial,i;
      initial=h(x);
      i=0;
      while((i<B)&&(strcmp(D[(initial+i)%B],x)!=0)&&
            (strcmp(D[(initial+i)%B],empty)!=0))i=i+1;
      return (initial+i)%B;
}

int locate1(ElementType x, Dictionary D){
      int initial,i;
      initial=h(x);
      i=0;
      while((i<B)&&(strcmp(D[(initial+i)%B],x)!=0)&&
            (strcmp(D[(initial+i)%B],empty)!=0)&&
            (strcmp(D[(initial+i)%B],deleted)!=0))i=i+1;
      return (initial+i)%B;
}

/*check if a member is in the dictionary*/
int member(ElementType x, Dictionary D){
      return strcmp(D[locate(x,D)],x)==0;     //so sanh 2 chuoi
}

/*insert a member into the dictionary*/
void insertdict(ElementType x, Dictionary D){
      int bucket;
      if(strcmp(D[locate(x,D)],x)!=0){
           bucket=locate1(x,D);         //se chen x vao bucket nay
           if((strcmp(D[bucket],empty)==0)||(strcpy(D[bucket],deleted)==0))
               D[bucket]=strdup(x);
           else printf("Error! Hash table is full");
      }
}
/*delete a member from the dictionary*/
void deletedict(ElementType x, Dictionary D){
      int bucket;
      bucket=locate(x,D);
      if(strcmp(D[bucket],x)==0)
           D[bucket]=strdup(deleted);
}
/*Enter a hash table*/
void read_hashtable(Dictionary D){
      char s[50];
      int i;

      printf("\nInput English words, enter to stop.\n");
      makenulldict(D);
      do{
           printf(" word=");gets(s);
```

```
            if(strcmp(s,"")!=0)
                insertdict(s,D);
      }while(strcmp(s,"")!=0);
}
/*print out the dict.*/
void print_dict(Dictionary D){
      int i;
      printf("\nDictionary:\n");
      for(i=0;i<B;i++)
         if((strcmp(D[i],empty)!=0)&&(strcmp(D[i],deleted)!=0))
           printf("%s\n",D[i]);
}
int main(){
      int x; char word[50];
      Dictionary D;
      read_hashtable(D);
      print_dict(D);

      printf("\nEnter a word to find it:");scanf("%s",word);
      if(member(word,D))
         printf("\nThis word is already in the dictionary.");
      else  printf("\nThis word has not been in the dictionary.");

      printf("\nEnter a word to delete it:");scanf("%s",word);
      deletedict(word,D);
      if(member(word,D))
         printf("\nThis word is still in the dictionary.");
      else printf("\nThis word is not in the dictionary.");
      print_dict(D);
      getch();
      return 0;
}
```

Chú ý: *strcmp và strdup nằm trong thư viện string.h.*

*int strcmp(const char *s1, const char *s2): so sánh 2 chuỗi s1 và s2.*

- *<0 nếu s1<s2*
- *==0 nếu s1==s2*
- *>0 nếu s1>s2*

*char *strdup(const char *s); cấp phát 1 vùng nhớ mới, rồi copy chuỗi s vào đó. Không gian được cấp phát dài strlen(s)+1 bytes.*

9) Viết chương trình cài đặt một tự điển dùng bảng băm mở để lưu các số nguyên. Dùng hàm băm h(x)=x%B, với B=5. Thực hiện các công việc sau:

- **Nhập vào 1 số nguyên. Tìm kiếm xem nó có trong tự điển hay không.**
- **Nhập vào 1 số nguyên. Xóa nó khỏi tự điển. Kiểm tra lại.**

```
/*BAM_MO.C*/
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#define B  5     /*number of buckets*/
typedef int ElementType;
typedef struct Node{
      ElementType Data;
      struct Node *next;
};

typedef struct Node *Position;
typedef Position Dictionary[B];
```

```
/*hash function*/
int h(ElementType x){
      return x%B;
}
/*create a empty hashtable*/
void MakeNullSet(Dictionary &D){
      int i;
      for(i=0;i<B;i++)
         D[i]=NULL;
}
/*check a memeber if it is in the dictionary*/
int Member(ElementType x, Dictionary D){
      Position p;
      int found=0;
      p=D[h(x)];
      while((p!=NULL)&&(!found))
            if(p->Data==x)found=1;
            else p=p->next;
      return found;
}

/*add a member into dictionary*/
void InsertSet(ElementType x, Dictionary &D){
      int Bucket;
      Position p;
      if(!Member(x,D)){
            Bucket=h(x);
            p=D[Bucket];
            D[Bucket]=(Node*)malloc(sizeof(Node));
            D[Bucket]->Data=x;
            D[Bucket]->next=p;
      }
}
/*delete a member from the dictionary*/
void DeleteSet(ElementType x, Dictionary &D){
      int Bucket, Done;
      Position p,q;
      Bucket=h(x);
      /*if x is in the header of the list*/
      if(D[Bucket]->Data==x){
            q=D[Bucket];
            D[Bucket]=D[Bucket]->next;
            free(q);
      }else{/*find x*/
            Done=0;
            p=D[Bucket];
            while((p->next!=NULL)&&(!Done))
                  if(p->next->Data==x)Done=1;
                  else p=p->next;

            /*x is found*/
            if(Done){
                  /*delete p->next*/
                  q=p->next;
                  p->next=q->next;
                  free(q);
            }
      }
}
```

```
/*Input data for a dictionary*/
void read_hashtable(Dictionary &D){
      int x;
      printf("Enter a list of integer numbers. -1 to stop.\n");
      MakeNullSet(D);
      do{   printf("x=");scanf("%d",&x);
            if(x!=-1)
                InsertSet(x,D);
      }while (x!=-1);
}
int main(){
      Dictionary D;
      int x;
      read_hashtable(D);

      printf("Enter a number to find it\n");
      printf("x=");scanf("%d",&x);

      if(Member(x,D))
          printf("%d is already in the dictionary.",x);
      else printf("%d has not been in the dictionary.",x);
      getch();
      return 0;
}
```

10) Viết chương trình cài đặt một cây tổng quát dùng con trỏ.

a)Viết khai báo để mỗi nút được lưu các thông tin:

- **Nhãn**
- **Con trái nhất**
- **Anh em ruột phải**
- **Cha**

b) Viết hàm create(v,r1,r2) để tạo một cây có gốc nhãn v, con trái nhất là r1 và anh em ruột phải là r2.

c) Gọi create bên trên để truyền vào 1 cây có nhiều nút.

d) Duyệt: tiền tự, trung tự, hậu tự.

```
/*CAYTQCONTRO.C*/
#include<conio.h>
#include<stdio.h>
#include<malloc.h>
#define NIL -1
typedef int datatype;

typedef struct node{
        datatype data;
        node*  leftmostchild;
        node*  rightsibling;
        node*  parent;

        };

typedef node* position;
typedef position tree;

//khoi tao cay rong
void makenull_tree(tree& T){
     T=NULL;
     }

//ktra cay rong
int empty_tree(tree T){
    return T==NULL;
}

//xac dinh nut cha
position parent(position n,tree T){
     if (empty_tree(T))
        return NULL;
     else return n->parent;
     }
//xd nhan cua nut
datatype label_node(position n,tree T){
         if(!(empty_tree(T))&&n!=NULL)
             return n->data;
     }

//xd nut goc
position root(tree T){
     if(!empty_tree(T)) return T;
     else return NULL;
    }
```

```
//ham xd con trai nhat cua 1 nut
position leftmostchild(tree T){
     if (T!=NULL ) return T->leftmostchild;
     else return NULL;
   }


//xd anh em ruot phai
position rightsibling(position n, tree T){
     if (n!=NULL )return n->rightsibling;
     else return NULL;}


//tao cay moi tu 2 cay co san
tree create(datatype v,tree r1,tree r2)
     {
         position n;
         n=(node*)malloc(sizeof(node));
         n->data=v;
         n->leftmostchild=r1;
         n->rightsibling=r2;
         n->parent=NULL;

         if(r1!=NULL )r1->parent=n;
         if(r2!=NULL)r2->parent=n;

         return n;
     }

//thu tuc duyet tien tu
void preorder(tree T){
          position p;
          printf("%5d",T->data);

          p=leftmostchild(T);
          while(p!=NULL)
            {
                preorder(p);
                p=rightsibling(p,T);
            }
          }

 //thu tuc duyet trung tu
void inorder(tree T){
          position p;

          p=leftmostchild(T);
          if(p!=NULL)inorder(p);

          printf("%5d",T->data);

          p=rightsibling(p,T);
          while(p!=NULL)
            {
                inorder(p);
                p=rightsibling(p,T);
            }
          }
```

```
//thu tuc duyet hau tu
void posorder(tree T){
          position p;

          p=leftmostchild(T);
          while(p!=NULL)
            {
                posorder(p);
                p=rightsibling(p,T);
            }

          printf("%5d",T->data);

         }

int main(){
    tree T;

  //              5
  //            / | \
  //           4  3  1

    T=create(5,  create(4,NULL,create(3,NULL, create(1,NULL,NULL))), NULL);
    printf("tien tu:");preorder(T);
    printf("\ntrung tu:");inorder(T);
    printf("\nhau tu:");posorder(T);

    getch();
    return 0;
 }
```

11) Một thủ tục duyệt theo mức cho cây nhị phân có thể được viết như sau:

Với các khai báo như sau:

```
//Khai báo cho cây
typedef int KeyType;
typedef struct Node{
    KeyType Key;
    struct Node *left,*right;

};
typedef struct Node* Tree;
typedef struct Node* Position;

//Cài đặt các phép toán của cây(sinh viên tự làm)
. . .

//Khai báo cho hàng
#define Maxlength 100
typedef Position ElementType;
//mỗi phần tử của hàng lưu 1 con trỏ trỏ tới 1 nút của cây
typedef struct{
    ElementType Elements[Maxlength];
    int Front, Rear;
}Queue;


//Cài đặt các phép toán cho hàng (sinh viên tự làm)
. . .
```

```
void LevelOrder(Tree T){
     Queue Q;
     Position P;

     MakeNull_Queue(Q);

     P=T;
     if(P!=NULL)EnQueue(P,Q);

     while(!Empty_Queue(Q))
     {
       P=Front(Q);
       printf("%5d",P->Key);
       DeQueue(Q);

       if(LeftChild(P)!=NULL)EnQueue(LeftChild(P),Q);
       if(RightChild(P)!=NULL)EnQueue(RightChild(P),Q);
     }//while
  }//void
```

// **so nut 2 con**

```
int nut2con(tree t)
{
   if((t==NULL)||(leftchild(t)==NULL)||(rightchild(t)==NULL)) return 0;
   else
      if(leftchild(t)!=NULL&&rightchild(t)!=NULL)
         return 1+nut2con(t->left)+nut2con(t->right);
}
```

12) Nhập vào một dãy số nguyên để lưu vào hàng ưu tiên. In ra theo thứ tự DeleleMin.

```
#include <stdio.h>
#include <conio.h>

#define MaxLength 100 //Đo dai mang rong
typedef int ElementType; //Kieu phan tu
typedef struct {
    ElementType Elements[MaxLength];
    int Last;
}PriorityQueue;

//Tao hàng uu tiên rong
void MAKENULL(PriorityQueue &A){
    A.Last=0;
}

int EMPTY(PriorityQueue A){
    return     A.Last==0;
    }
//chen x vao hang - dung vi tri cua no trong cay thu tu tung phan
void INSERT(ElementType X,PriorityQueue &A){
    int i;
    ElementType temp;
    if (A.Last==MaxLength) printf(" Loi : Mang day");
    else
    {
          A.Last++;
          A.Elements[A.Last]=X;
```

```
          i=A.Last;
          while ((i>0) && (A.Elements[i]<A.Elements[(i-1)/2]))
          {
                temp=A.Elements[i];
                A.Elements[i]=A.Elements[(i-1)/ 2];
                A.Elements[(i-1) /2]=temp;
                i= (i-1) / 2;
          }
    }
}

//xoa phan tu co do uu tien be nhat -phan tu goc cua cay
ElementType DELETEMIN(PriorityQueue &A)
{   int i,j, done;
    ElementType temp, minimum;

    if (A.Last==0) printf(" Hang dang rong");
    else{
          minimum=A.Elements[0];

          A.Elements[0]=A.Elements[A.Last];
          A.Last--;

          i=0;
          done=0;
          while ((i<=A.Last/2) && (!done) )
          {
                if((2*i+1==A.Last) || (A.Elements[2*i+1]<A.Elements[2*i+2]))
                            j=2*i+1;
                else j=2*i+2;
                if (A.Elements[i]>A.Elements[j])
                {
                      temp=A.Elements[i];
                      A.Elements[i]=A.Elements[j];
                      A.Elements[j]=temp;
                      i=j;
                }
                else done=1;
          }
    return minimum;
    }
}

main(){
       PriorityQueue A;
       ElementType x;

       MAKENULL(A);
       //nhap vao 1 day cac so nguyen, -1 de dung
       printf("Nhap vao 1 day so de dua vao hang, -1 de dung:\n");
       do{
             printf("x=");scanf("%d",&x);
             INSERT(x,A);
       }while(x!=-1);

       printf("\nHang theo thu tu DeleleMin: ");
       while(!EMPTY(A)){
              printf("%5d",DELETEMIN(A));
              }
```

```
getch();
}
```

HẾT