

```
////////////////////////////////////  
// Một phần chương trình sắp xếp  
// Thay vì phải mỗi lần chạy các thuật toán sắp xếp ta phải nhập giá trị cho mảng,  
// Các thủ tục cho phép đọc dữ liệu từ file text, file này có 02 dòng; dòng đầu tiên có  
// 01 con số cho biết số phần tử của mảng, dòng thứ 02 là các giá trị của các phần tử  
// ví dụ: mảng có 03 phần tử, có các giá trị là 12, 4 và 6  
// 3  
// 12 4 6
```

```
////////////////////////////////////  
#include<stdio.h>  
#include<stdlib.h>  
#include<conio.h>  
void readFromFile(char*name,int *&a,int &n);  
int *copyArray(int *a,int n);  
void printData(int *a,int n);  
void freeMemory(int *&a);  
  
void main(){  
int *a;  
int n;  
clrscr();  
readFromFile("c:\\temp\\c\\2.cpp",a,n);  
printData(a,n);  
getch();  
freeMemory(a);  
}  
void readFromFile(char*name,int *&a,int &n){  
FILE *f;  
if((f=fopen(name,"rt"))==NULL){  
printf("\n Can not read File !");  
exit(0);  
}  
fscanf(f,"%d",&n);  
if(n<1){  
printf("\n Data Error !");  
exit(0);  
}  
a = new int[n];  
if(a==NULL){  
printf("\n Not enough memory !");  
exit(0);
```

```
    }

for(int i=0;i<n;i++)fscanf(f,"%d",&a[i]);
}

int *copyArray(int *a,int n){
int *b = new int[n];
if(b==NULL){
    printf("\n Not enough memory !");
    exit(0);
}
for(int i=0;i<n;b[i]=a[i],i++);
return b;
}

void printData(int *a,int n){
for(int i=0;i<n;printf("%d\t",a[i],i++);
}

void freeMemory(int *&a){
delete[]a;
}
```

```
////////////////////////////////////
// Phần cơ bản của lập trình các thuật toán sắp xếp, tìm kiếm theo hướng đối tượng
////////////////////////////////////
```

```
#include<iostream.h>
#include<stdlib.h>
#include<conio.h>
#define max 500
class Array{
private:
    int *a;
    int num;
    int lsearch(int);
    int bsearch(int);
    int checkOrder();
    int isOk();
    void randomData();
    int randomCheck(int);
    int llsearch(int,int);
```

```
        void sort2(){ }
public:
    void sort1();
    int BSearch(int);
    int LSearch(int x){return lsearch(x);}
    Array();
    Array(int);
    Array(int *,int);
    ~Array(){delete []a;num=0;}
    void printData();
};
void Array::sort1()
{
    if(!isOk()||checkOrder())randomData();
    cout<<"\nData befor sort !\n";
    printData();
    sort2();
    cout<<"\nAfter Sorted !\n";
    printData();
}
void Array::printData()
{
    if(num>100)
    {
        cout<<"\nPress Any Key To Continous\n";
        getch();
    }

    if(!isOk())
    {
        cout<<"\n\naNot Have Data !\n";
        return;
    }
    cout<<"\n\n";
    for(int i=0;i<num;i++)
        cout<<a[i]<<"\t";

}
int Array::checkOrder()
{
    int i=0;
```

```
        while(i<num-1&& a[i]<=a[i+1])i++;
        if(i==num-1)return 1;
        i=0;
        while(i<num-1&& a[i]>=a[i+1])i++;
        if(i==num-1)return 1;
        return 0;
    }
int Array::isOk()
{
    if(a&&num>0)return 1;
    return 0;
}
int Array::randomCheck(int n)
{
    randomize();
    if(n<1)return random(max);
    int data;
    do{
        data = random(max);
    }while(!lsearch(data,n));
    return data;
}
int Array::lsearch(int d,int n)
{
    if(n>num)
    {
        cout<<"\n\la Error !\n";
        exit(0);
    }

    for(int i=0;i<n;i++)
        if(a[i]==d)return 1;
    return 0;
}
void Array::randomData()
{
    randomize();
    do{
        num = random(max-200);
    }while(num<10);
    a = new int[num];
}
```

```
        if(a==NULL)
            {
                cout<<"\n\a Not Enough Memory !\n";
                exit(0);
            }

        for(int i=0;i<num;i++)
            a[i]=randomCheck(i);
    }
int Array::BSearch(int x)
    {
        if(!checkOrder())return -2;
        return bsearch(x);
    }
int Array::lsearch(int x)
    {
        for(int i=0;i<num;i++)
            if(a[i]==x)return i;
        return -1;
    }
int Array::bsearch(int x)
    {
        int l=0,r=num-1;
        do{
            int mid = (l+r)/2;
            if(a[mid]==x)return mid;
            if(a[mid]<x)l=mid+1;
            else r=mid-1;
        }while(l<=r);
        return -1;
    }
Array::Array()
    {
        a = NULL;
        num =0;
    }
Array::Array(int n)
    {
        num = n;
        a = new int[num];
        if(a==NULL)
```

```
        {
            cout<<"\n\a Not Enough Memory !\n";
            exit(0);
        }
    }
Array::Array(int *b,int n)
    {
        num = n;
        a = new int[num];
        if(a==NULL)
            {
                cout<<"\n\a Not Enough Memory !\n";
                exit(0);
            }
        for(int i=0;i<n;i++)a[i]=b[i];
    }
void main()
{
    clrscr();
    Array t;
    int b[]={ 13,9,3,78};
    t.sort1();
    Array g(b,4);
    g.sort1();
    getch();
}
```

```
////////////////////////////////////
// Chương trình dùng DSLK đơn (kiểu hướng đối tượng) để quản lý nhiều phần tử
// có cấu trúc khác nhau thông qua dữ liệu kiểu con trỏ void để quản lý
// ngày 22 tháng 04 năm 2004 - @ by Phạm Thế Bảo
// Khoa Toán – Tin học trường ĐHKHTN Tp. Hồ Chí Minh
////////////////////////////////////
```

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

```
struct NV{
    int luong;
    char *name;
```

```
};

class Student{
    private:
        int code;
        char *name;
    public:
        Student(int c,char* st){code=c;name = st;}
        ~Student(){code=-1;}
        int getCode(){return code;}
        char *getName(){return name;}
};

class TagNode{
friend class LinkedList;
    private:
        void *data;
        char kind;
        TagNode *Next;
    public:
        TagNode();
        TagNode(void *,char);
        ~TagNode();
};

typedef TagNode *Node;
class LinkedList{
    private:
        Node Head,Tail;
    public:
        LinkedList();
        ~LinkedList();
        int insertNodeAtHead(void *d,char t);
        void print();
};

////////////////////////////////////
// code của các phương thức trong 02 lớp TagNode và LinkedList
////////////////////////////////////
TagNode::TagNode(){
    kind=-1;
    data = Next = NULL;
};
```

```
    }
TagNode::TagNode(void *a,char t){
    kind = t;
    data = a;
    Next = NULL;
}
TagNode::~~TagNode(){
    kind =-1;
    if(Next) delete Next;
}
LinkedList::LinkedList(){
    Head = Tail = NULL;
}
LinkedList::~~LinkedList(){
    if(Head)delete Head;
}
int LinkedList::insertNodeAtHead(void *d,char t){
    Node p = new TagNode(d,t);
    if(!p)return -1;
    if(!Head)Head = Tail = p;
    else{
        p->Next = Head;
        Head = p;
    }
    return 1;
}
void LinkedList::print(){
    Node p = Head;
    while(p){
        switch(p->kind){
            case 0: // kiểu dữ liệu là số nguyên
                int t = (int) (void *)p->data;
                printf("\n%d",t);
                break;
            case 1: // kiểu dữ liệu là chuỗi ký tự
                char *tt = (char*)(void *)p->data;
                printf("\n%s",tt);
                break;
            case 2:// kiểu dữ liệu là đối tượng Student
                Student *t1 = (Student*)(void *)p->data;
                printf("\ncode = %d name =%s",
```

```
                t1->getCode(),t1->getName());
                break;
            case 3:// kiểu dữ liệu là một struct NV
                NV *nv1 = (NV*)(void *)p->data;
                printf("\nLuong = %d Ten =%s",nv1->luong,
                    nv1->name);
                break;
            case 4:// kiểu dữ liệu là ký tự
                char cc = (char)(void *)p->data;
                printf("%c",cc);
        }
        p = p->Next;
    }
}

////////////////////////////////////////////////////////////////
// Hàm Main
// Tạo và xuất dữ liệu của 01 phần tử của DSLK có các kiểu dữ liệu là:
//           1. 01 số nguyên
//           2. 01 chuỗi ký tự
//           3. 01 struct NV
//           4. 01 đối tượng Student
//           5. 01 ký tự
////////////////////////////////////////////////////////////////

int main(){
    clrscr();
    LinkedList ll;
    void *c;
    for(int i=12;i<20;i++){
        c = (void *) i;
        ll.insertNodeAtHead(c,0);
    }

    char *temp;
    printf("Nhap 01 chuoai ky tu:");
    gets(temp);
    c = (void *) temp;
    ll.insertNodeAtHead(c,1);

    Student *temp1= new Student(1243,"I like coco");
    c = (void *)temp1;
    ll.insertNodeAtHead(c,2);
}
```

```
NV *a= new NV;;
a->luong = 300;
printf("Nhap ho ten:");
gets(a->name);
c = (void *) a;
ll.insertNodeAtHead(c,3);

for(i=65;i<100;i++){
    c = (void *) i;
    ll.insertNodeAtHead(c,4);
}

ll.print();
getch();
return 1;
}
```

```
////////////////////////////////////
// Đoạn Chương trình để vẽ cây nhị phân tìm kiếm
// ngày 12 tháng 01 năm 2004 - @ by Phạm Thế Bảo
// Khoa Toán – Tin học trường ĐHKHTN Tp. Hồ Chí Minh
////////////////////////////////////
```

```
#include<graphics.h>
```

```
#define DX 20
```

```
#define DH 30
```

```
char s[5]="0000";
```

```
int CellH, CellW;
```

```
void Init(int &H, int &W)
```

```
{
```

```
    H = textheight(s)+8;
```

```
    W = textwidth(s) + 6;
```

```
}
```

```
void drawTree(Node T, int x, int y, int width)
```

```
{
```

```
    if(!T)return;
```

```
    rectangle(x-CellW/2, y-CellH/2, x+CellW/2, y+CellH/2);
```

```
    sprintf(s, "%d", T->Data);
```

```
    outtextxy(x, y, s);
    if (T->Left)
    {
        line(x, y+CellH/2, x-width, y+CellH/2+DH);
        drawTree(T->Left, x-width, y+CellH+DH, width-DX);
    }
    if (T->Right)
    {
        line(x, y+CellH/2, x+width, y+CellH/2+DH);
        drawTree(T->Right, x+width, y+CellH+DH, width-DX);
    }
}
void drawTree(BSTree T)
{
    int graphdriver=DETECT ,graphmode,errorcode;
    initgraph(&graphdriver,&graphmode,"c:\\bc\\bgi");
    errorcode = graphresult();
    if (errorcode != grOk) /* an error occurred */
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1); /* terminate with an error code */
    }
    Init(CellH, CellW);
    settxtjjustify(CENTER_TEXT, CENTER_TEXT);
    // draw BSTree
    drawTree(T.Root, getmaxx()/2, 10, 60);
    getch();
    closegraph();
}
```
