Nội dung

* Các thuật toán sắp xếp cơ bản
  + Sắp xếp lựa chọn
  + Sắp xếp chèn
  + Sắp xếp nổi bọt

Một số ứng dụng của bài toán sắp xếp

VD 1. Tổ chức danh bạ điện thoại (sắp xếp theo tên hoặc cả họ tên), tra cứu danh bạ theo họ tên

================================

Bài toán sắp xếp

Thuật toán sắp xếp chèn

* ý tưởng: Chèn 1 phần tử vào danh sách đã có thứ tự sao cho sau khi chèn danh sách vẫn đúng thứ tự  
  1,4,5,7,23 cần chèn 17 vào  
  Để tìm vị trí chèn ta cần tìm tuần tự trong danh sách --> O(n)  
  1,4,5,7,17,23
* Xuất phát từ danh sách ban đầu chỉ có 1 phần tử (đã có thứ tự)  
  Chèn lần lượt n-1 phần tử vào danh sách đã có thứ tự, áp dụng ý tưởng thuật toán chèn ở trên
* Sau khi chèn đủ n-1 phần tử còn lại --> thu được danh sách đã sắp xếp

Nếu cài đặt trên mảng, việc chèn phần tử vào ==> phải dịch mảng

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 12 | 17 | 21 |  |  |  |

Chèn 3 --> các phần tử 5,12,17,21 phải dịch sang phải 1 vi trí để lấy chỗ chèn 3

==> chi phí thực hiện sắp xếp chèn trên mảng cao!

Thuật toán sắp xếp chèn sẽ được sử dụng nhiều trên danh sách liên kết đơn vì KHÔNG cần phải dịch phần tử

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>    typedef struct node  {  char hoten[30];  float diem;  struct node \*pNext;  } NODE;    NODE\* insert(NODE \*pHead, NODE\* ptr)  {  if(pHead==NULL) pHead=ptr;  else // tìm vi tri chen vao giua/cuoi danh sach  {  if(pHead->diem >= ptr->diem) //phan tu moi them dung dau danh sach cu  {  ptr->pNext=pHead;  pHead=ptr; // cap nhat lai dau  }  else  {  NODE \*preQ=pHead; // con tro toi phan tu ngay truoc vi tri chen  NODE \*q=pHead->pNext; // la con tro toi vi tri chen  while(q!=NULL && q->diem < ptr->diem)  {  preQ=q;  q=q->pNext;  }  ptr->pNext=q; // chen ptr vao sau preQ  preQ->pNext=ptr;  }  }  return pHead;  }    NODE\* insertionSort(NODE \*head)  {  if(head==NULL) return NULL;  //Ngat danh sach thanh 2 phan  // phan dau dang tro boi head chi co 1 phan tu  // phan con lai tro boi secondList la n-1 phan tu se duoc chen lai  NODE \*secondList = head->pNext, \*p;  head->pNext = NULL;  while(NULL!=secondList)  {  p = secondList;  secondList = secondList->pNext;  p->pNext = NULL;  head = insert(head,p);  }  return head;  }      NODE \*createNode(char\* hoten, float diem)  {  NODE \*newNode= (NODE\*)malloc(sizeof(NODE));  strcpy(newNode->hoten,hoten);  newNode->diem = diem;  newNode->pNext = NULL;  return newNode;  }      void clearBuffer()  {  int c;  while ((c = getchar()) != '\n' && c != EOF) { }  }  // tao danh sach nhap tu ban phim    NODE\* createList()  {  int size;  printf("So luong phan tu: ");  scanf("%d",&size);  char hoten[30];  float diem;  NODE \*head=NULL, \*newNode;  for(int i=0; i<size;i++)  {  printf("Hoten: ");  clearBuffer();  //gets(hoten);  if (fgets(hoten, sizeof hoten, stdin)) {  hoten[strcspn(hoten, "\n")] = '\0';  }  printf("Diem: ");  scanf("%f",&diem);  if(head==NULL) head = createNode(hoten,diem);  else {  newNode = createNode(hoten,diem);  newNode->pNext = head;  head = newNode;  }  }  return head;  }    // in ra danh sach  void printList(NODE \*head)  {  printf("Noi dung danh sach hien tai:\n");  while(NULL!=head)  {  printf("Ho ten: %s\t", head->hoten);  printf("Diem: %.2f\n", head->diem);  head = head->pNext;  }  printf("\n");  }    int main()  {  NODE\* head=NULL;  head = createList();  printf("Danh sach ban dau:\n");  printList(head);  head = insertionSort(head);  printf("Danh sach sau khi sap xep:\n");  printList(head);  return 0;  } |

~~Bài 1. Sửa lại để có thể nhập vào nội dung danh sách từ file~~

|  |
| --- |
| 4  Nguyen Van A  4.5  Le Thi B  7.5  Tran Van C  8  Dao Van A  3.5 |

Bài 2. Ghi lại danh sách đã sắp xếp ra file

Bài 3. Thêm chức năng bổ sung thêm người vào danh sách

Bài 4. Tạo menu dạng

1. Nạp danh sách từ file
2. Thêm người vào danh sách hiện tại
3. Sắp xếp danh sách hiện tại theo điểm (giảm dần hoặc tăng dần)
4. Ghi danh sách hiện tại ra file

Bài 5. Tách ra phần tên khỏi họ tên và sắp xếp danh sách theo họ tên thứ tự ABC tăng dần

|  |
| --- |
| // chi la tham chieu, chi duoc doc, KHONG duoc sua doi  // ham tra ve vi tri bat dau cua ten trong xau hoten chu KHONG tao ra vung nho moi  char\* getName(char\* hoten)  {  // la vi tri dau cach trong cuoi cung  int pos=strlen(hoten);  while(pos>=0 && hoten[pos]!=' ') pos--;  if(pos<0)// hoten ko co dau cach trong, va do la ten  return hoten;  else // tach ra phan ten tu pos+1  return &hoten[pos+1];  } |
| char hoten[35] = "Nguyen Van Dung";  printf("Hoten %s, ten %s\n",hoten,getName(hoten)); |

|  |
| --- |
| typedef struct node  {  char hoten[30];  char ten[30]; // bo nho phu cho phan ten  float diem;  struct node \*pNext;  } NODE;  // chi la tham chieu, chi duoc doc, KHONG duoc sua doi  // ham tra ve vi tri bat dau cua ten trong xau hoten chu KHONG tao ra vung nho moi  char\* getName(char\* hoten)  {  // la vi tri dau cach trong cuoi cung  int pos=strlen(hoten);  while(pos>=0 && hoten[pos]!=' ') pos--;  if(pos<0)// hoten ko co dau cach trong, va do la ten  return hoten;  else // tach ra phan ten tu pos+1  return &hoten[pos+1];  }  NODE \*createNode(char\* hoten, float diem)  {  NODE \*newNode= (NODE\*)malloc(sizeof(NODE));  strcpy(newNode->hoten,hoten);  strcpy(newNode->ten,getName(hoten));  newNode->diem = diem;  newNode->pNext = NULL;  return newNode;  } |

Hàm chuẩn hóa xâu

|  |
| --- |
| // bo ky tu \r\n o cuoi xau neu doc tu file  char\* chuanHoaXau(char\* str)  {  int pos = 0;  for(pos=0;pos<strlen(str);pos++)  if(str[pos]=='\r' || str[pos]=='\n'){  str[pos]='\0';  break;  }  return str;  } |
|  |