**Danh sách kế tiếp**

**#define max 50**

**typedef struct List{**

**int elems[max];**

**int size; };**

**void init(struct List \*L){**

**(\*L).size = 0; }**

**void nhaplist(struct List \*L){**

**int n;**

**printf("Nhap n: ");**

**scanf("%d",&n);**

**if(n<0 || n>max){**

**printf("So luong pt ko hop le.\n");**

**return; }**

**printf("Nhap %d phan tu:\n",n);**

**for(int i=0;i<n;i++){**

**printf("Phan tu thu %d: ",i+1);**

**scanf("%d",&(\*L).elems[i]); }**

**(\*L).size = n;}**

**void inlist(List L){**

**printf("Danh sach cac phan tu: ");**

**for(int i=0;i<L.size;i++){**

**printf("%d ",L.elems[i]); }**

**printf("\n");}**

**void chenxvaok(int x,int k,struct List \*L){**

**if(L->size >= max){**

**printf("Danh sach day, ko the chen.\n");**

**return; }**

**if(k<1 || k> L->size+1) printf("Khong the chen.\n");**

**else{**

**for(int i=L->size;i >= k;i--){**

**L->elems[i] = L->elems[i-1]; }**

**L->elems[k-1] = x;**

**L->size++;**

**printf("Mang sau khi chen %d vao vi tri %d la: ",x,k);**

**for(int i=0;i< L->size;i++){**

**printf("%d ",L->elems[i]); }**

**printf("\n"); } }**

**void xoaphantuvitrik(int k,struct List \*L){**

**if(k<1 || k>L->size+1) printf("Khong the xoa.\n");**

**else{**

**for(int i =k-1;i<L->size;i++){**

**L->elems[i] = L->elems[i+1]; }**

**L->size--;**

**printf("Mang sau khi xoa 1 phan tu o vi tri %d la: ",k);**

**for(int i=0;i<L->size;i++){**

**printf("%d ",L->elems[i]); } } }**

**void timvitrix(int x,struct List \*L) {**

**int timthay = 0;**

**for(int i=0;i<L->size;i++){**

**if(L->elems[i] == x){**

**printf("Vi tri dau tien cua %d la: %d\n",x,i);**

**timthay = 1;**

**break; } }**

**for(int i=0;i < L->size; i++) {**

**if (L->elems[i] == x) {**

**timthay = i; } }**

**if (!timthay) {**

**printf("Khong tim thay phan tu %d trong danh sach\n", x); }**

**else {**

**printf("Vi tri cuoi cung cua %d la: %d\n", x,timthay);**

**printf("Tat ca vi tri tim thay %d la: ",x);**

**for(int i=0;i < L->size; i++) {**

**if (L->elems[i] == x) {**

**printf("%d ",i); } } } }**

**void ptchanleamduong(struct List \*L){**

**int chan=0,le=0,am=0,duong=0;**

**for(int i=0;i<L->size;i++){**

**if(L->elems[i] % 2 == 0){**

**chan++;**

**} else {**

**le++; }**

**if(L->elems[i] > 0){**

**duong++;**

**} else {**

**am++; }**

**} }**

**int trungbinhduong(struct List \*L){**

**int soptduong = 0;**

**int tong = 0;**

**for(int i =0;i<L->size;i++){**

**if(L->elems[i] > 0){**

**tong = tong + L->elems[i];**

**soptduong++;}**

**}**

**float tbc = 0;**

**if(soptduong > 0){**

**tbc = (float)tong/soptduong;**

**printf("Trung binh cua cac phan tu duong: %g",tbc);**

**} else {**

**return 0;**

**}**

**printf("\n");}**

**void tachmangcon(struct List L,struct List \*listchan,struct List \*listle,struct List \*listam,struct List \*listduong){**

**init(listchan);**

**init(listle);**

**init(listam);**

**init(listduong);**

**for(int i=0;i<L.size;i++){**

**if(L.elems[i] % 2 == 0){**

**listchan->elems[listchan->size] = L.elems[i];**

**listchan->size++;**

**} else {**

**listle->elems[listle->size] = L.elems[i];**

**listle->size++;**

**}**

**if(L.elems <0){**

**listam->elems[listam->size] = L.elems[i];**

**listam->size++;**

**} else if (L.elems >0){**

**listduong->elems[listduong->size] = L.elems[i];**

**listduong->size++;**

**}**

**} }**

**int search(List &L,int x){**

**for(int i=0;i<L.size;i++){**

**if(L.elems[i] ==x) return i+1;**

**else return 0;**

**} }**

**void xoaptgiatrix(int x, List &L){**

**int i,j;**

**for(i=0;i<L.size;i++){**

**if(L.elems[i] == x){**

**for(j =i;j<L.size-1;j++){**

**L.elems[j] = L.elems[j+1];**

**}**

**L.size--;**

**i--;**

**} }**

**inlist(L); }**

**Sinh viên kế tiếp**

**typedef struct SV {**

**char MaSv[10];**

**char HoTen[50];**

**int NamSinh;**

**char Lop[10];**

**float DiemTB;**

**};**

**typedef struct List{**

**struct SV elems[max];**

**int size;**

**};**

**void init(struct List \*L) {**

**L->size = 0;**

**}**

**void nhapList(struct List \*L) {**

**int n, i;**

**printf("Nhap so luong sinh vien (n <= %d): ", max);**

**scanf("%d", &n);**

**if (n <= 0 || n > max) {**

**printf("So luong sinh vien khong hop le!\n");**

**return;**

**}**

**getchar();**

**printf("Nhap thong tin cho %d sinh vien:\n", n);**

**for (i = 0; i < n; i++) {**

**printf("Sinh vien thu %d:\n", i + 1);**

**printf("Ma SV: ");**

**gets(L->elems[i].MaSv);**

**printf("Ho Ten: ");**

**gets(L->elems[i].HoTen);**

**printf("Nam Sinh: ");**

**scanf("%d", &L->elems[i].NamSinh);**

**getchar();**

**printf("Lop: ");**

**gets(L->elems[i].Lop);**

**printf("Diem TB: ");**

**scanf("%f", &L->elems[i].DiemTB);**

**getchar();**

**}**

**L->size = n;**

**}**

**void inList(struct List L) {**

**int i;**

**printf("%-10s%-20s%-10s%-10s%-10s\n", "Ma SV", "Ho Ten", "Nam Sinh", "Lop", "Diem TB");**

**for (i = 0; i < L.size; i++) {**

**printf("%-10s%-20s%-10d%-10s%-10.2f\n", L.elems[i].MaSv, L.elems[i].HoTen, L.elems[i].NamSinh, L.elems[i].Lop, L.elems[i].DiemTB);**

**}**

**}**

**void themSV(struct List \*L, struct SV SVmoi, int k) {**

**int i;**

**if (L->size >= max) {**

**printf("Danh sach da day, khong the them sinh vien moi!\n");**

**return;**

**}**

**if (k < 0 || k > L->size) {**

**printf("Vi tri chen khong hop le!\n");**

**return;**

**}**

**for (i = L->size; i > k-1; i--) {**

**L->elems[i] = L->elems[i - 1];**

**}**

**L->elems[k-1] = SVmoi;**

**L->size++;**

**}**

**void themSVvaok(struct List \*L) {**

**struct SV SVmoi;**

**int k;**

**printf("Nhap thong tin sinh vien can them:\n");**

**printf("Ma SV: ");**

**gets(SVmoi.MaSv);**

**printf("Ho Ten: ");**

**gets(SVmoi.HoTen);**

**printf("Nam Sinh: ");**

**scanf("%d", &SVmoi.NamSinh);**

**getchar();**

**printf("Lop: ");**

**gets(SVmoi.Lop);**

**printf("Diem TB: ");**

**scanf("%f", &SVmoi.DiemTB);**

**getchar();**

**printf("Nhap vi tri chen: ");**

**scanf("%d", &k);**

**themSV(L, SVmoi, k);**

**printf("Danh sach sau khi chen:\n");**

**inList(\*L);**

**}**

**void xoavitrik(struct List \*L,int k){**

**printf("Vi tri xoa: ");**

**scanf("%d",&k);**

**if(k<0 || k >= L->size){**

**printf("Vi tri xoa khong hop le!\n");**

**return;**

**}**

**for(int i=k-1;i<L->size-1;i++){**

**L->elems[i] = L->elems[i+1];**

**}**

**L->size--;**

**printf("Danh sach sau khi xoa: \n");**

**inList(\*L);**

**}**

**int timdautien(struct List \*L,char \*ten){**

**for(int i=0;i<L->size;i++){**

**if(strcmp(L->elems[i].HoTen, ten) == 0){**

**return i+1;**

**}**

**}**

**return -1;**

**}**

**int timcuoicung(struct List \*L,char \*ten){**

**int cuoicung = -1;**

**for(int i=0;i<L->size;i++){**

**if(strcmp(L->elems[i].HoTen, ten) == 0){**

**cuoicung = i+1;**

**}**

**}**

**return cuoicung;**

**}**

**void timtatca(struct List \*L,char \*ten){**

**int dem = 0;**

**printf("Tim thay sinh vien co ten %s o vi tri thu: ",ten);**

**for (int i = 0; i < L->size; i++){**

**if (strcmp(L->elems[i].HoTen, ten) == 0) {**

**printf("%d ", i+1);**

**dem++;**

**}**

**} if(dem == 0){**

**printf("Khong tim thay sinh vien co ten %c\n", ten);**

**}**

**}**

**void demsvtren55(struct List \*L){**

**int dem=0;**

**for(int i=0;i<L->size;i++){**

**if(L->elems[i].DiemTB > 5.5){**

**dem++;**

**}**

**}**

**printf("So sinh vien co dtb tren 5.5 la: %d\n",dem);**

**}**

**void tachsvtheolop(struct List L, struct List Llop[]){**

**int dem=0;**

**for(int i=0;i<L.size;i++){**

**int tim =0;**

**for(int j=0;j<dem;j++){**

**if(strcmp(Llop[j].elems[0].Lop, L.elems[i].Lop)==0){**

**Llop[j].elems[Llop[j].size] = L.elems[i];**

**Llop[j].size++;**

**tim =1;**

**break;**

**}**

**}**

**if(!tim){**

**init(&Llop[dem]);**

**Llop[dem].elems[0] = L.elems[i];**

**Llop[dem].size++;**

**dem++;**

**}**

**}**

**}**

**Danh sách liên kết**

**typedef struct Node{**

**int Data;**

**Node\* next;**

**};**

**typedef struct Node\* plist;**

**void init(plist\* L){**

**\*L = NULL;}**

**int length(plist L){**

**Node\* M = L;**

**int dem =0;**

**while(M != NULL){**

**M = M->next;**

**dem++;**

**}**

**return dem;}**

**Node\* Make\_Node(Node \*P,int x){**

**P = (Node\*)malloc(sizeof(Node));**

**P->Data = x;**

**P->next = NULL;**

**return P;}**

**void themdau(plist \*L,int x){**

**Node \*P= Make\_Node(P,x);**

**P->next = \*L;**

**\*L=P;}**

**void themcuoi(plist \*L,int x){**

**Node \*P = Make\_Node(P,x);**

**Node \*M = \*L;**

**if(M == NULL){**

**\*L = P;**

**return; }**

**while(M->next != NULL)**

**M = M-> next;**

**M->next =P; }**

**void themvaovitrik(plist \*L,int x,int k){**

**if(k<1 || k>length(\*L)+1){**

**printf("Khong the them!");**

**return;}**

**if(k == 1){**

**themdau(L,x);**

**return; }**

**Node \*P = Make\_Node(P,x);**

**Node\* M = \*L;**

**int dem =1;**

**while(M->next != NULL && dem != k-1){**

**M = M->next;**

**dem++; }**

**P->next = M->next;**

**M->next = P; }**

**void xoadau(plist \*L){**

**\*L = (\*L)->next;**

**free(\*L);}**

**void xoacuoi(plist \*L){**

**if(\*L == NULL){**

**printf("Khong the chen.\n");**

**return;}**

**if((\*L)->next == NULL){**

**free(\*L);**

**\*L == NULL;**

**return; }**

**Node \*M =\*L;**

**while(M->next->next != NULL){**

**M = M->next; }**

**free(M->next);**

**M->next = NULL;}**

**void xoavtk(plist \*L,int k){**

**if(\*L ==NULL || k<1 || k>length(\*L)){**

**printf("khong the xoa.\n");**

**return; }**

**if(k == 1){**

**xoadau(L);**

**return; }**

**Node \*M=\*L;**

**int dem=1;**

**while(M != NULL && dem != k-1){**

**M = M->next;**

**dem++; }**

**Node\* temp = M->next;**

**M->next = temp->next;**

**free(temp);}**

**void nhapdl(plist \*L){**

**int n,data;**

**printf("Nhap so luong phan tu: ");**

**scanf("%d",&n);**

**for(int i=0;i<n;i++){**

**printf("Nhap gia tri phan tu thu %d: ",i+1);**

**scanf("%d",&data);**

**themcuoi(L,data); } }**

**void inds(plist L){**

**Node \*M=L;**

**printf("Danh sach: ");**

**while(M != NULL){**

**printf("%3d",M->Data);**

**M=M->next;**

**}**

**printf("\n"); }**

**int timpttrongds(plist L,int x){**

**Node\* M=L;**

**int vitri=0;**

**int tim =0;**

**printf("Tim thay phan tu %d tai vi tri: ",x);**

**while(M != NULL){**

**vitri++;**

**if(M->Data == x){**

**printf("%3d",vitri);**

**tim =1;**

**}**

**M = M->next;**

**}**

**printf("\n");**

**if(!tim){**

**printf("Khong tim thay %d\n",x);**

**}**

**return vitri; }**

**void timdautimcuoi(plist L,int x){**

**Node \*M=L;**

**int vitridau = -1;**

**int vitricuoi = -1;**

**int vitri = 0;**

**while(M!= NULL){**

**vitri++;**

**if(M->Data == x){**

**if(vitridau == -1){**

**vitridau = vitri;**

**}**

**vitricuoi = vitri; }**

**M=M->next; }**

**if(vitridau != -1){**

**printf("Vi tri dau tien cua %d: %d\n",x,vitridau);**

**printf("Vi tri cuoi cua %d: %d\n",x,vitricuoi);**

**}else{**

**printf("Khong tim thay phan tu %d\n",x);**

**} }**

**Sinh viên liên kết**

**typedef struct SV {**

**char MaSv[10];**

**char HoTen[50];**

**int NamSinh;**

**char Lop[10];**

**float DiemTB;**

**}SV;**

**typedef struct Node{**

**SV \*data;**

**struct Node \*next;**

**}Node;**

**typedef struct List{**

**Node \*dau;**

**int soluong;**

**}List;**

**void init(List \*L){**

**L->soluong=0;**

**L->dau = NULL;**

**}**

**SV \*taoSV() {**

**SV \*sv = (SV \*)malloc(sizeof(SV));**

**if (sv == NULL) {**

**printf("Loi: Khong the cap phat bo nho cho sinh vien moi.\n");**

**exit(1);**

**}**

**printf("Ma SV: ");**

**scanf("%s", sv->MaSv);**

**printf("Ho Ten: ");**

**scanf(" %[^\n]", sv->HoTen);**

**printf("Nam Sinh: ");**

**scanf("%d", &sv->NamSinh);**

**printf("Lop: ");**

**scanf("%s", sv->Lop);**

**printf("Diem TB: ");**

**scanf("%f", &sv->DiemTB);**

**return sv;**

**}**

**Node \*Make\_Node(SV \*x) {**

**Node \*P = (Node \*)malloc(sizeof(Node));**

**P->data = x;**

**P->next = NULL;**

**return P;**

**}**

**void themSinhVienVaoDS(List \*L, SV \*sv) {**

**Node \*M = Make\_Node(sv);**

**if (L->dau == NULL) {**

**L->dau = M;**

**} else {**

**Node \*current = L->dau;**

**while (current->next != NULL) {**

**current = current->next;**

**}**

**current->next = M;**

**}**

**L->soluong++;**

**}**

**void inList(List \*L) {**

**if (L->soluong == 0) {**

**printf("Danh sach rong.\n");**

**return;**

**}**

**printf("%-10s%-20s%-10s%-10s%-10s\n", "Ma SV", "Ho Ten", "Nam Sinh", "Lop", "Diem TB");**

**Node \*current = L->dau;**

**while (current != NULL) {**

**SV \*sv = current->data;**

**printf("%-10s%-20s%-10d%-10s%-10.2f\n", sv->MaSv, sv->HoTen, sv->NamSinh, sv->Lop, sv->DiemTB);**

**current = current->next;**

**}**

**}**

**void themDauDS(List \*L, SV \*sv) {**

**Node \*M = Make\_Node(sv);**

**if (L->dau == NULL) {**

**L->dau = M;**

**} else {**

**M->next = L->dau;**

**L->dau = M;**

**}**

**L->soluong++;**

**}**

**void themCuoiDS(List \*L, SV \*sv) {**

**Node \*M = Make\_Node(sv);**

**if (L->dau == NULL) {**

**L->dau = M;**

**} else {**

**Node \*current = L->dau;**

**while (current->next != NULL) {**

**current = current->next;**

**}**

**current->next = M;**

**}**

**L->soluong++;**

**}**

**void themVaoK(List \*L, SV \*sv, int k) {**

**if (k < 1 || k > L->soluong + 1) {**

**printf("Vi tri them khong hop le.\n");**

**return;**

**}**

**Node \*newNode = Make\_Node(sv);**

**if (k == 1) {**

**newNode->next = L->dau;**

**L->dau = newNode;**

**} else {**

**Node \*current = L->dau;**

**int i;**

**for (i = 1; i < k - 1; i++) {**

**current = current->next;**

**}**

**newNode->next = current->next;**

**current->next = newNode;**

**}**

**L->soluong++;**

**}**

**void xoadau(struct List \*L) {**

**if (L->soluong == 0) {**

**printf("Danh sach rong, khong the xoa!\n");**

**return;**

**}**

**Node \*temp = L->dau;**

**L->dau = L->dau->next;**

**free(temp->data);**

**free(temp);**

**L->soluong--;**

**printf("Danh sach sau khi xoa dau:\n");**

**}**

**void xoacuoi(struct List \*L) {**

**if (L->soluong == 0) {**

**printf("Danh sach rong, khong the xoa!\n");**

**return;**

**}**

**if (L->soluong == 1) {**

**free(L->dau->data);**

**free(L->dau);**

**L->dau = NULL;**

**} else {**

**Node \*current = L->dau;**

**while (current->next->next != NULL) {**

**current = current->next;**

**}**

**free(current->next->data);**

**free(current->next);**

**current->next = NULL;**

**}**

**L->soluong--;**

**printf("Danh sach sau khi xoa cuoi:\n");**

**}**

**void xoavitrik(struct List \*L, int k) {**

**if (k < 1 || k > L->soluong) {**

**printf("Vi tri xoa khong hop le!\n");**

**return;**

**}**

**if (k == 1) {**

**xoadau(L);**

**} else {**

**Node \*current = L->dau;**

**int i;**

**for (i = 1; i < k - 1; i++) {**

**current = current->next;**

**}**

**Node \*temp = current->next;**

**current->next = temp->next;**

**free(temp->data);**

**free(temp);**

**L->soluong--;**

**printf("Danh sach sau khi xoa vi tri %d:\n", k);**

**}**

**}**

**int timSinhVien(List \*L, char \*ten, int vitri[]) {**

**int tim = 0;**

**int dem = 0;**

**Node \*current = L->dau;**

**while (current != NULL) {**

**SV \*sv = current->data;**

**if (strcmp(sv->HoTen, ten) == 0) {**

**tim = 1;**

**vitri[dem++] = L->soluong - L->soluong + 1;**

**}**

**current = current->next;**

**}**

**return tim;**

**}**

**int dtbhon5\_5(List \*L, float diemTB) {**

**int dem = 0;**

**Node \*current = L->dau;**

**while (current != NULL) {**

**SV \*sv = current->data;**

**if (sv->DiemTB > diemTB) {**

**dem++;**

**}**

**current = current->next;**

**}**

**return dem;**

**}**

**void tachsvtheolop(List \*L, List \*dsLop[], int \*soLuongLop) {**

**Node \*current = L->dau;**

**while (current != NULL) {**

**SV \*sv = current->data;**

**int found = 0;**

**for (int i = 0; i < \*soLuongLop; i++) {**

**if (strcmp(sv->Lop, dsLop[i]->dau->data->Lop) == 0) {**

**themCuoiDS(dsLop[i], sv);**

**found = 1;**

**break;**

**}**

**}**

**if (!found) {**

**List \*newList = (List \*)malloc(sizeof(List));**

**init(newList);**

**themCuoiDS(newList, sv);**

**dsLop[\*soLuongLop] = newList;**

**(\*soLuongLop)++;**

**}**

**current = current->next;**

**}**

**}**

**Ngăn xếp kế tiếp**

**#define MAXSIZE 100**

**struct Stack {**

**int data[MAXSIZE];**

**int top;**

**};**

**void init(struct Stack \*S) {**

**S->top = -1; }**

**int isEmpty(struct Stack \*S) {**

**return (S->top == -1);**

**}**

**int isFull(struct Stack \*S) {**

**return (S->top == MAXSIZE - 1);**

**}**

**void push(struct Stack \*S, int x) {**

**if (isFull(S)) {**

**printf("Ngan xep day, khong the them phan tu moi.\n");**

**return;**

**}**

**S->top++;**

**S->data[S->top] = x;**

**}**

**void pop(struct Stack \*S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong, khong the xoa phan tu.\n");**

**return;**

**}**

**S->top--;**

**}**

**void nhapStack(struct Stack \*S) {**

**int n, x;**

**printf("Nhap so luong phan tu can them vao ngan xep: ");**

**scanf("%d", &n);**

**for (int i = 0; i < n; i++) {**

**printf("Nhap phan tu thu %d: ", i + 1);**

**scanf("%d", &x);**

**push(S, x);**

**}**

**}**

**void inStack(struct Stack \*S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong.\n");**

**return;**

**}**

**printf("Danh sach phan tu trong ngan xep:\n");**

**for (int i = 0; i <= S->top; i++) {**

**printf("%d ", S->data[i]);**

**}**

**printf("\n");**

**}**

**int peek(struct Stack \*S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong, khong co phan tu de lay ra.\n");**

**return -1;**

**}**

**return S->data[S->top];**

**}**

**void hehai(int hemuoi) {**

**struct Stack hehaiStack;**

**init(&hehaiStack);**

**while (hemuoi > 0) {**

**push(&hehaiStack, hemuoi % 2);**

**hemuoi /= 2;**

**}**

**printf("So nhi phan: ");**

**while (!isEmpty(&hehaiStack)) {**

**printf("%d", peek(&hehaiStack));**

**pop(&hehaiStack);**

**}**

**printf("\n");**

**}**

**void hetam(int hemuoi) {**

**struct Stack hetamStack;**

**init(&hetamStack);**

**while (hemuoi > 0) {**

**push(&hetamStack, hemuoi % 8);**

**hemuoi /= 8;**

**}**

**printf("So bat phan: ");**

**while (!isEmpty(&hetamStack)) {**

**printf("%d", peek(&hetamStack));**

**pop(&hetamStack);**

**}**

**printf("\n");**

**}**

**void hemuoisau(int hemuoi) {**

**struct Stack hemuoisauStack;**

**init(&hemuoisauStack);**

**while (hemuoi > 0) {**

**int phandu = hemuoi % 16;**

**if (phandu < 10) {**

**push(&hemuoisauStack, phandu);**

**} else {**

**push(&hemuoisauStack, phandu - 10 + 'A');**

**}**

**hemuoi /= 16;**

**}**

**printf("So hex: ");**

**while (!isEmpty(&hemuoisauStack)) {**

**printf("%c", peek(&hemuoisauStack));**

**pop(&hemuoisauStack);**

**}**

**printf("\n");**

**}**

**void themcuoi(struct Stack \*S, int x) {**

**struct Stack M;**

**init(&M);**

**while (!isEmpty(S)) {**

**push(&M, peek(S));**

**pop(S);**

**}**

**push(S, x);**

**while (!isEmpty(&M)) {**

**push(S, peek(&M));**

**pop(&M);**

**}**

**}**

**void themxvaok(struct Stack \*S, int x, int k) {**

**if (k < 1 || k > S->top + 2) {**

**printf("Vi tri k khong hop le.\n");**

**return;**

**}**

**struct Stack temp;**

**init(&temp);**

**for (int i = 1; i < k; i++) {**

**push(&temp, peek(S));**

**pop(S);**

**}**

**push(S, x);**

**while (!isEmpty(&temp)) {**

**push(S, peek(&temp));**

**pop(&temp);**

**}**

**}**

**void xoaday(struct Stack \*S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong, khong co phan tu de xoa.\n");**

**return;**

**}**

**struct Stack temp;**

**init(&temp);**

**while (S->top > 0) {**

**push(&temp, peek(S));**

**pop(S);**

**}**

**pop(S);**

**while (!isEmpty(&temp)) {**

**push(S, peek(&temp));**

**pop(&temp);**

**}**

**}**

**void xoavitrik(struct Stack \*S, int k) {**

**if (k < 1 || k > S->top + 1) {**

**printf("Vi tri k khong hop le.\n");**

**return;**

**}**

**struct Stack temp;**

**init(&temp);**

**for (int i = 1; i < k; i++) {**

**push(&temp, peek(S));**

**pop(S);**

**}**

**pop(S);**

**while (!isEmpty(&temp)) {**

**push(S, peek(&temp));**

**pop(&temp);**

**}**

**}**

**Ngăn xếp liên kết**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

**struct Stack {**

**struct Node\* top;**

**};**

**void init(struct Stack\* S) {**

**S->top = NULL;**

**}**

**int isEmpty(struct Stack\* S) {**

**return (S->top == NULL);**

**}**

**void push(struct Stack\* S, int x) {**

**struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));**

**if (newNode == NULL) {**

**printf("Khong du bo nho.\n");**

**return;**

**}**

**newNode->data = x;**

**newNode->next = S->top;**

**S->top = newNode;**

**}**

**void pop(struct Stack\* S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong, khong the xoa phan tu.\n");**

**return;**

**}**

**struct Node\* temp = S->top;**

**S->top = S->top->next;**

**free(temp);**

**}**

**void nhapStack(struct Stack\* S) {**

**int n, x;**

**printf("Nhap so luong phan tu can them vao ngan xep: ");**

**scanf("%d", &n);**

**for (int i = 0; i < n; i++) {**

**printf("Nhap phan tu thu %d: ", i + 1);**

**scanf("%d", &x);**

**push(S, x);**

**}**

**}**

**void inStack(struct Stack\* S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong.\n");**

**return;**

**}**

**printf("Danh sach phan tu trong ngan xep:\n");**

**struct Node\* current = S->top;**

**while (current != NULL) {**

**printf("%d ", current->data);**

**current = current->next;**

**}**

**printf("\n");**

**}**

**//Các hàm khác giữ nguyên nội dung chỉ cần thay đổi có pháp S.top thành S->top**

**Sinh viên ngăn xếp kế tiếp**

**struct SinhVien {**

**char MaSV[20];**

**char HoTen[50];**

**int NamSinh;**

**char Lop[20];**

**float DiemTB;**

**};**

**struct Stack {**

**struct SinhVien data[MAX];**

**int top;**

**};**

**void init(struct Stack \*S) {**

**S->top = -1;**

**}**

**int isEmpty(struct Stack \*S) {**

**return S->top == -1;**

**}**

**int isFull(struct Stack \*S) {**

**return S->top == MAX - 1;**

**}**

**void Push(struct Stack \*S,struct SinhVien sv) {**

**if (isFull(S)) {**

**printf("Ngan xep da day, khong the them sinh vien.\n");**

**return;**

**}**

**S->data[++S->top]=sv;**

**}**

**void Pop(struct Stack \*S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong, khong the xoa sinh vien.\n");**

**return;**

**}**

**S->top--;**

**}**

**void themsvvaoday(struct Stack \*S, struct SinhVien sv) {**

**if (isFull(S)) {**

**printf("Ngan xep da day, khong the them sinh vien vao day.\n");**

**return;**

**}**

**struct Stack tempStack;**

**init(&tempStack);**

**while (!isEmpty(S)) {**

**Push(&tempStack, S->data[S->top]);**

**Pop(S);**

**}**

**Push(S, sv);**

**while (!isEmpty(&tempStack)) {**

**Push(S, tempStack.data[tempStack.top]);**

**Pop(&tempStack);**

**}**

**}**

**void themsvvaok(struct Stack \*S, struct SinhVien sv, int k) {**

**if (isFull(S)) {**

**printf("Ngan xep da day, khong the them sinh vien vao vi tri %d.\n", k);**

**return;**

**}**

**struct Stack tempStack;**

**init(&tempStack);**

**while (!isEmpty(S)) {**

**Push(&tempStack, S->data[S->top]);**

**Pop(S);**

**}**

**int count = 0;**

**while (!isEmpty(&tempStack)) {**

**if (count == k) {**

**Push(S, sv);**

**}**

**Push(S, tempStack.data[tempStack.top]);**

**Pop(&tempStack);**

**count++;**

**}**

**if (count == k) {**

**Push(S, sv);**

**}**

**}**

**void xoaday(struct Stack \*S) {**

**if (isEmpty(S)) {**

**printf("Ngan xep rong, khong the xoa sinh vien tai day.\n");**

**return;**

**}**

**struct Stack tempStack;**

**init(&tempStack);**

**while (!isEmpty(S)) {**

**Push(&tempStack, S->data[S->top]);**

**Pop(S);**

**}**

**Pop(&tempStack);**

**while (!isEmpty(&tempStack)) {**

**Push(S, tempStack.data[tempStack.top]);**

**Pop(&tempStack);**

**}**

**}**

**void xoavtk(struct Stack \*S, int k) {**

**if (isEmpty(S) || k < 1 || k > S->top + 1) {**

**printf("Khong the xoa sinh vien tai vi tri %d.\n", k);**

**return;**

**}**

**struct Stack tempStack;**

**init(&tempStack);**

**while (!isEmpty(S)) {**

**Push(&tempStack, S->data[S->top]);**

**Pop(S);**

**}**

**int count = 0;**

**while (!isEmpty(&tempStack)) {**

**if (count != k - 1) {**

**Push(S, tempStack.data[tempStack.top]);**

**}**

**Pop(&tempStack);**

**count++;**

**}**

**while (!isEmpty(&tempStack)) {**

**Push(S, tempStack.data[tempStack.top]);**

**Pop(&tempStack);**

**}**

**}**

**void nhapdanhsach(struct SinhVien \*sv) {**

**printf("Nhap ma SV: ");**

**scanf("%s", sv->MaSV);**

**printf("Nhap ho ten: ");**

**scanf("%s", sv->HoTen);**

**printf("Nhap nam sinh: ");**

**scanf("%d", &sv->NamSinh);**

**printf("Nhap lop: ");**

**scanf("%s", sv->Lop);**

**printf("Nhap diem TB: ");**

**scanf("%f", &sv->DiemTB);**

**}**

**void inStack(struct Stack \*S) {**

**printf("Danh sach sinh vien trong ngan xep:\n");**

**printf("%-10s%-20s%-10s%-10s%-10s\n", "Ma SV", "Ho Ten", "Nam Sinh", "Lop", "Diem TB");**

**for (int i = S->top; i >= 0; i--) {**

**struct SinhVien sv = S->data[i];**

**printf("%-10s%-20s%-10d%-10s%-10.2f\n", sv.MaSV, sv.HoTen, sv.NamSinh, sv.Lop, sv.DiemTB);**

**}**

**}**

**Hàng đợi**

**#define n 100**

**typedef struct Queue{**

**int elem[n];**

**int count;**

**int front, rear;**

**}Queue;**

**void init(Queue \*Q){**

**Q->count=0;**

**Q->front=0;**

**Q->rear= -1;**

**}**

**int isEmpty(Queue \*Q){**

**return (Q->count==0);**

**}**

**int isFull(Queue \*Q){**

**return (Q->count == n);**

**}**

**void enQueue(Queue \*Q, int x) {**

**if (isFull(Q)) {**

**printf("Hang doi day, khong the them phan tu moi.\n");**

**return;**

**}**

**Q->rear = (Q->rear +1)%n;**

**Q->elem[Q->rear] = x;**

**Q->count++;**

**}**

**void deQueue(Queue \*Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa phan tu.\n");**

**return;**

**}**

**int x = Q->elem[Q->front];**

**Q->front = (Q->front + 1) % n;**

**Q->count--;**

**}**

**void input(Queue \*Q) {**

**int m, item;**

**printf("Nhap so luong phan tu: ");**

**scanf("%d", &m);**

**for (int i = 0; i < m; i++) {**

**printf("Nhap phan tu thu %d: ", i + 1);**

**scanf("%d", &item);**

**enQueue(Q, item);**

**}**

**}**

**void display(Queue \*Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong.\n");**

**return;**

**}**

**printf("Du lieu trong hang doi: ");**

**int i = Q->front;**

**for(int j=0;j<Q->count;j++){**

**printf("%d ",Q->elem[i]);**

**i = (i+1)%n;**

**}**

**printf("\n");**

**}**

**void themvaodau(Queue \*Q, int x) {**

**if (isFull(Q)) {**

**printf("Hang doi day, khong the them pha tu moi.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**while (!isEmpty(Q)) {**

**int item = Q->elem[Q->front];**

**deQueue(Q);**

**enQueue(&tempQ, item);**

**}**

**enQueue(Q, x);**

**while (!isEmpty(&tempQ)) {**

**int item = tempQ.elem[tempQ.front];**

**deQueue(&tempQ);**

**enQueue(Q, item);**

**}**

**}**

**void themxvaok(Queue \*Q, int x, int k) {**

**if (isFull(Q) || k < 1 || k > Q->count + 1) {**

**printf("Vi tri khong hop le hoac hang doi day.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**for (int i = 1; i < k; i++) {**

**int item = Q->elem[Q->front];**

**deQueue(Q);**

**enQueue(&tempQ, item);**

**}**

**enQueue(Q, x);**

**while (!isEmpty(&tempQ)) {**

**int item = tempQ.elem[tempQ.front];**

**deQueue(&tempQ);**

**enQueue(Q, item);**

**}**

**}**

**void xoacuoi(Queue \*Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa phan tu.\n");**

**return;**

**}**

**int item = Q->elem[Q->rear];**

**Q->rear=(Q->rear -1 +n)%n;**

**Q->count--;**

**}**

**void xoavitrik(Queue \*Q, int k) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa phan tu.\n");**

**return;**

**}**

**if (k < 1 || k > Q->count) {**

**printf("Vi tri k khong hop le.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**for (int i = 1; i < k; i++) {**

**int item = Q->elem[Q->front];**

**deQueue(Q);**

**enQueue(&tempQ, item);**

**}**

**deQueue(Q);**

**while (!isEmpty(&tempQ)) {**

**int item = tempQ.elem[tempQ.front];**

**deQueue(&tempQ);**

**enQueue(Q, item);**

**} }**

**Hàng đợi liên kết**

**typedef struct Node {**

**int data;**

**struct Node\* next;**

**} Node;**

**typedef struct Queue {**

**Node\* front;**

**Node\* rear;**

**int count;**

**} Queue;**

**void init(Queue\* Q) {**

**Q->front = NULL;**

**Q->rear = NULL;**

**Q->count = 0;**

**}**

**int isEmpty(Queue\* Q) {**

**return (Q->count == 0);**

**}**

**void enQueue(Queue\* Q, int x) {**

**Node\* newNode = (Node\*)malloc(sizeof(Node));**

**if (newNode == NULL) {**

**printf("Khong du bo nho.\n");**

**return;**

**}**

**newNode->data = x;**

**newNode->next = NULL;**

**if (isEmpty(Q)) {**

**Q->front = newNode;**

**Q->rear = newNode;**

**} else {**

**Q->rear->next = newNode;**

**Q->rear = newNode;**

**}**

**Q->count++;**

**}**

**void deQueue(Queue\* Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa phan tu.\n");**

**return;**

**}**

**Node\* temp = Q->front;**

**Q->front = Q->front->next;**

**free(temp);**

**Q->count--;**

**}**

**void input(Queue\* Q) {**

**int m, item;**

**printf("Nhap so luong phan tu: ");**

**scanf("%d", &m);**

**for (int i = 0; i < m; i++) {**

**printf("Nhap phan tu thu %d: ", i + 1);**

**scanf("%d", &item);**

**enQueue(Q, item);**

**}**

**}**

**void display(Queue\* Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong.\n");**

**return;**

**}**

**printf("Du lieu trong hang doi: ");**

**Node\* current = Q->front;**

**while (current != NULL) {**

**printf("%d ", current->data);**

**current = current->next;**

**}**

**printf("\n");**

**}**

**//Các hàm khác giữ nguyên nội dung chỉ cần thay đổi có pháp Q.front thành Q.->front**

**Sinh viên hàng đợi kế tiếp**

**#define n 100**

**typedef struct {**

**char MaSv[20];**

**char HoTen[50];**

**int NamSinh;**

**char Lop[20];**

**float DiemTB;**

**} SinhVien;**

**typedef struct {**

**SinhVien elem[n];**

**int front, rear, count;**

**} Queue;**

**void init(Queue \*Q) {**

**Q->count = 0;**

**Q->front = 0;**

**Q->rear = -1;**

**}**

**int isEmpty(Queue \*Q) {**

**return (Q->count == 0);**

**}**

**int isFull(Queue \*Q) {**

**return (Q->count == n);**

**}**

**void enQueue(Queue \*Q, SinhVien sv) {**

**if (isFull(Q)) {**

**printf("Hang doi day, khong the them sinh vien moi.\n");**

**return;**

**}**

**Q->rear = (Q->rear + 1) % n;**

**Q->elem[Q->rear] = sv;**

**Q->count++;**

**}**

**void deQueue(Queue \*Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa.\n");**

**return;**

**}**

**SinhVien sv = Q->elem[Q->front];**

**Q->front = (Q->front + 1) % n;**

**Q->count--;**

**}**

**SinhVien nhapSinhVien() {**

**SinhVien sv;**

**printf("Nhap thong tin sinh vien:\n");**

**printf("Ma SV: ");**

**scanf("%s", sv.MaSv);**

**printf("Ho ten: ");**

**scanf(" %[^\n]", sv.HoTen);**

**printf("Nam sinh: ");**

**scanf("%d", &sv.NamSinh);**

**printf("Lop: ");**

**scanf(" %[^\n]", sv.Lop);**

**printf("Diem TB: ");**

**scanf("%f", &sv.DiemTB);**

**return sv;**

**}**

**void display(Queue \*Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong.\n");**

**return;**

**}**

**printf("Du lieu trong hang doi:\n");**

**printf("%-10s%-20s%-10s%-10s%-10s\n", "Ma SV", "Ho Ten", "Nam Sinh", "Lop", "Diem TB");**

**int i = Q->front;**

**for (int j = 0; j <Q->count; j++) {**

**SinhVien sv = Q->elem[i];**

**printf("%-10s%-20s%-10d%-10s%-10.2f\n", sv.MaSv, sv.HoTen, sv.NamSinh, sv.Lop, sv.DiemTB);**

**i = (i + 1) % n;**

**}**

**}**

**void themdau(Queue \*Q, SinhVien sv) {**

**if (isFull(Q)) {**

**printf("Hang doi day, khong the them sinh vien moi.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**int i = Q->front;**

**for (int j = 0; j < Q->count; j++) {**

**enQueue(&tempQ, Q->elem[i]);**

**i = (i + 1) % n;**

**}**

**init(Q);**

**enQueue(Q, sv);**

**i = tempQ.front;**

**for (int j = 0; j < tempQ.count; j++) {**

**enQueue(Q, tempQ.elem[i]);**

**i = (i + 1) % n;**

**}**

**}**

**void themsvvaok(Queue \*Q, SinhVien sv, int k) {**

**if (isFull(Q)) {**

**printf("Hang doi day, không the them sinh vien moi.\n");**

**return;**

**}**

**if (k < 1 || k > Q->count + 1) {**

**printf("Vi tri k khong hop le.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**int i = Q->front;**

**for (int j = 0; j < Q->count; j++) {**

**if (j == k - 1) {**

**enQueue(&tempQ, sv);**

**}**

**enQueue(&tempQ, Q->elem[i]);**

**i = (i + 1) % n;**

**}**

**if (k == Q->count + 1) {**

**enQueue(&tempQ, sv);**

**}**

**init(Q);**

**i = tempQ.front;**

**for (int j = 0; j < tempQ.count; j++) {**

**enQueue(Q, tempQ.elem[i]);**

**i = (i + 1) % n;**

**}**

**}**

**void xoacuoi(Queue \*Q) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa sinh vien.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**int i = Q->front;**

**for (int j = 0; j < Q->count - 1; j++) {**

**enQueue(&tempQ, Q->elem[i]);**

**i = (i + 1) % n;**

**}**

**init(Q);**

**i = tempQ.front;**

**for (int j = 0; j < tempQ.count; j++) {**

**enQueue(Q, tempQ.elem[i]);**

**i = (i + 1) % n;**

**}**

**}**

**void xoasvvtk(Queue \*Q, int k) {**

**if (isEmpty(Q)) {**

**printf("Hang doi rong, khong the xoa sinh vien.\n");**

**return;**

**}**

**if (k < 1 || k > Q->count) {**

**printf("Vi tri k khong hop le.\n");**

**return;**

**}**

**Queue tempQ;**

**init(&tempQ);**

**int i = Q->front;**

**for (int j = 0; j < Q->count; j++) {**

**if (j != k - 1) {**

**enQueue(&tempQ, Q->elem[i]);**

**}**

**i = (i + 1) % n;**

**}**

**init(Q);**

**i = tempQ.front;**

**for (int j = 0; j < tempQ.count; j++) {**

**enQueue(Q, tempQ.elem[i]);**

**i = (i + 1) % n;**

**}**

**}**

**Cây np**

**typedef struct Node{**

**int key;**

**struct Node\* left;**

**struct Node\* right;**

**}Node;**

**typedef struct Node\* Btree;**

**Btree T;**

**void makeNullTree(Btree \*T){**

**(\*T)= NULL;**

**}**

**int emptytree(Btree T){**

**return T==NULL;**

**}**

**Btree leftchild(Btree T){**

**if(T != NULL) return T->left;**

**else return NULL;**

**}**

**Btree rightchild(Btree T){**

**if(T != NULL) return T->right;**

**else return NULL;**

**}**

**Btree findMin(Btree root) {**

**while (root->left != NULL) {**

**root = root->left;**

**}**

**return root;**

**}**

**int nutla(Btree T){**

**if(T != NULL){**

**return (leftchild(T)==NULL)&&(rightchild(T)==NULL);**

**}else{**

**return 0;**

**}**

**}**

**int sonut(Btree T){**

**if(emptytree(T)) return 0;**

**else return 1+sonut(leftchild(T))+sonut(rightchild(T));**

**}**

**Btree taocay(int x,Btree l,Btree r){**

**Btree N;**

**N = (Node\*)malloc(sizeof(Node));**

**N->key = x;**

**N->left = l;**

**N->right = r;**

**return N;**

**}**

**Btree search(Btree T, int x){**

**if (T == NULL || T->key == x) {**

**return T;**

**}**

**if (x < T->key) {**

**return search(T->left, x);**

**} else {**

**return search(T->right, x);**

**}**

**}**

**Btree themnutx(Btree T,int x){**

**if (T == NULL) {**

**return taocay(x, NULL, NULL);**

**}**

**if (x < T->key) {**

**T->left = themnutx(T->left, x);**

**} else if (x > T->key) {**

**T->right = themnutx(T->right, x);**

**}**

**return T;**

**}**

**Btree nhapdl(int n){**

**Btree P = NULL;**

**int x;**

**printf("Nhap gia tri cho cac nut: ");**

**for(int i=0;i<n;i++){**

**scanf("%d",&x);**

**P = themnutx(P,x);**

**}**

**return P;**

**}**

**int xoanut(Btree &T,int data){**

**if(T != NULL){**

**if(T->key == data){**

**if(T->left == NULL && T->right == NULL){**

**T = NULL;**

**return 0;**

**}**

**if(T->left != NULL && T->right ==NULL){**

**T = T->left;**

**return 0;**

**}**

**if(T->left == NULL && T->right != NULL){**

**T = T->right;**

**return 0;**

**}**

**if(T->left != NULL && T->right !=NULL){**

**Btree temp = T->left;**

**while(temp->right->right!=NULL){**

**temp = temp->right;**

**}**

**temp->right->left = T->left;**

**temp->right->right = T->right;**

**T = T->right;**

**temp->right =NULL;**

**return 0;**

**}**

**}**

**else{**

**xoanut(T->left,data);**

**xoanut(T->right,data);**

**}**

**}**

**}**

**void duyettruoc(Btree T){**

**if(T != NULL){**

**printf("%d ", T->key);**

**duyettruoc(T->left);**

**duyettruoc(T->right);**

**}**

**}**

**void duyetgiua(Btree T){**

**if(T != NULL){**

**duyetgiua(T->left);**

**printf("%d ", T->key);**

**duyetgiua(T->right);**

**}**

**}**

**void duyetsau(Btree T){**

**if(T != NULL){**

**duyetsau(T->left);**

**duyetsau(T->right);**

**printf("%d ",T->key);**

**}**

**}**

**Btree timcontrai(Btree T, int value) {**

**Btree node = search(T, value);**

**if (node != NULL) {**

**return leftchild(node);**

**} else {**

**return NULL;**

**}**

**}**

**Btree timconphai(Btree T, int value) {**

**Btree node = search(T, value);**

**if (node != NULL) {**

**return rightchild(node);**

**} else {**

**return NULL;**

**}**

**}**

**Btree themdinh(Btree T, int value) {**

**if (T == NULL) {**

**return taocay(value, NULL, NULL);**

**}**

**if (value < T->key) {**

**T->left = themdinh(T->left, value);**

**} else if (value > T->key) {**

**T->right = themdinh(T->right, value);**

**}**

**return T;**

**}**

**Btree deleteValue(Btree root, int value) {**

**if (root == NULL) {**

**return root;**

**}**

**if (value < root->key) {**

**root->left = deleteValue(root->left, value);**

**} else if (value > root->key) {**

**root->right = deleteValue(root->right, value);**

**} else {**

**if (root->left == NULL) {**

**Btree temp = root->right;**

**free(root);**

**return temp;**

**} else if (root->right == NULL) {**

**Btree temp = root->left;**

**free(root);**

**return temp;**

**}**

**Btree temp = findMin(root->right);**

**root->key = temp->key;**

**root->right = deleteValue(root->right, temp->key);**

**}**

**return root;**

**}**