DANH SÁCH LIÊN KẾT ĐƠN

(1) Lý thuyết:

- Xem bài giảng CHƯƠNG 4 (slide 1-20) và giáo trình trang 49-68.

(2) Bài tập tại lớp:

- Viết các chương trình theo các ví dụ trong slide bài giảng và giáo trình chương 4.

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<conio.h>
#include<malloc.h>

//Khai báo cấu trúc 1 nút trong danh sách liên kết đơn
struct node
{
	int info; //chứa nội dung của nút
	node *next; //con trỏ chỉ nút kế tiếp trong d/s
};

//Khai báo kiểu con trỏ chỉ đến nút
typedef node *NODEPTR;

//Cấp phát biến động làm 1 nút cho d/s
NODEPTR GetNode()
{
	NODEPTR p;
	p = (NODEPTR)malloc(sizeof(node));
	return p;
}

//Giải phóng biến động đã cấp phát trước đó
void FreeNode(NODEPTR p)
{
	free(p);
}

//Khởi động dslk
void Initialize(NODEPTR *plist)
{
	*plist = NULL;
}

//Kiểm tra d/s có bị rỗng hay không
int Empty(NODEPTR plist)
{
	return (plist == NULL ? 1 : 0);
}
```

```
//Xác định số nút có trong dslk
int ListSize(NODEPTR plist)
{
	int count=0;
	NODEPTR p = plist;
	while (p != NULL)
	{
		count++;
		p = p->next;
	}
	return count;
}

//Xác định con trỏ của nút thứ i (i = 0,1,2,...) trong dslk
NODEPTR NodePointer(NODEPTR plist, int i)
{
	if (i < 0)
		return NULL;
	else
	{
		NODEPTR p = plist;
		int pos = 0;
		while (p != NULL && pos < i)
		{
			p = p->next;
			pos++;
		}
		return p;
	}
}

//Xác định vị trí nút p trong dslk
int Position(NODEPTR plist, NODEPTR p)
{
	NODEPTR q = plist;
	int pos = 0;
	while (q != NULL && q != p)
	{
		q = q->next;
		pos++;
	}
	if (q != NULL)
		return pos;//vị trí của nút p trong d/s
	else
		return -1;//nút p không tồn tại
}
```

```
//Xác định nút trước của nút p trong dslk
NODEPTR PreNode(NODEPTR plist, NODEPTR p)
{
	if (p == plist)
		return NULL;
	else
	{
		NODEPTR q = plist;
		//Di chuyển q đến nút trước nút p
		while (q != NULL && q->next != p)
			q = q->next;
		return q;
	}
}

//Thêm nút có nội dung x vào đầu dslk
void Push(NODEPTR &plist, int x)
{
	NODEPTR p = GetNode();
	p->info = x;
	p->next = plist;
	plist = p;
}

//Thêm một nút có nội dung x ngay sau nút p
void InsAfter(NODEPTR p, int x)
{
	if (p == NULL)
		printf("Nut khong ton tai.\n");
	else
	{
		NODEPTR q = GetNode();
		q->info = x;
		q->next = p->next;
		p->next = q;
	}
}

//Xoá nút đầu trong dslk
void Pop(NODEPTR &plist)
{
	if (Empty(plist))
		printf("Danh sach bi rong\n");
	else
	{
		NODEPTR p = plist;
		int x = p->info;
		plist = p->next;
		p->next = NULL;
		FreeNode(p);
		printf("\nDa xoa nut dau co gia tri la %d\n", x);
	}
}
```

```
//Xoá nút ngay sau nút p trong dslk
void DelAfter(NODEPTR p)
{
	if (p == NULL || p->next == NULL) //Nếu nút p không tồn tại hoặc là nút cuối trong dslk
		printf("Khong xoa nut duoc\n");
	else
	{
		NODEPTR q = p->next;
		int x = q->info;
		p->next = q->next;
		q->next = NULL;
		FreeNode(q);
		printf("\nDa xoa nut sau co gia tri la %d\n", x);
	}
}

//Duyệt danh sách liên kết
void Traverse(NODEPTR plist)
{
	if (Empty(plist))
		printf("Danh sach bi rong\n");
	else
	{
		NODEPTR p = plist;
		while (p != NULL)
		{
			printf("%d\t", p->info);
			p = p->next;
		}
		printf("\n");
	}
}

//Tìm nút có nội dung x trong danh sách liên kết
NODEPTR Search(NODEPTR plist, int x)
{
	NODEPTR p = plist;
	while (p != NULL && p->info != x)
		p = p->next;
	return p;
}
```

```
//Sắp xếp danh sách liên kết theo thứ tự tăng dần của trường info
void Sort(NODEPTR *plist)
{
	NODEPTR p, q;
	int temp;
	for (p = *plist; p->next != NULL; p = p->next)
	{
		for (q = p->next; q != NULL; q = q->next)
		{
			if (p->info > q->info)
			{
				temp = p->info;
				p->info = q->info;
				q->info = temp;
			}
		}
	}
}

//Thêm nút có nội dung x trên danh sách liên kết đã có thứ tự
void Place(NODEPTR *plist, int x)
{
	NODEPTR p, q = NULL;
	//Duyệt d/s để tìm nút q ngay trước nút cần thêm
	for (p = *plist; p != NULL && x > p->info; p = p->next)
		q = p;
	if (q==NULL)
		Push(*plist, x);//gọi hàm thêm nút x vào đầu dslk
	else
		InsAfter(q, x);//gọi hàm thêm nút x vào sau nút q
}

//Xoá danh sách liên kết
void ClearList(NODEPTR *plist)
{
	NODEPTR p = *plist, q = NULL;
	while (p != NULL) //trong khi chưa đến cuối d/s
	{
		q = p;//xác định nút hiện hành
		p = p->next;//di chuyển đến nút kế tiếp
		FreeNode(q);//giải phóng nút hiện hành
	}
	Initialize(plist);//khởi động lại ds rỗng
}
```

```
void main()
{
	NODEPTR plist;
	Initialize(&plist);
	int chon;
	do
	{
		printf("\n-----CHUONG TRINH XU LY DANH SACH LIEN KET DON-----\n");
		printf("1. Them 1 nut x vao dau danh sach\n");
		printf("2. Them 1 nut x sau nut p trong danh sach\n");
		printf("3. Xoa nut dau danh sach\n");
		printf("4. Xoa nut sau nut p trong danh sach\n");
		printf("5. Duyet danh sach\n");
		printf("6. Tim nut x trong danh sach\n");
		printf("7. Sap xep danh sach tang dan theo truong info\n");
		printf("8. Them nut x vao d/s da duoc sap thu tu tang dan theo truong info\n");
		printf("9. Xoa dslk\n");
		printf("0. Thoat CT\n");
		printf("-------------------------------------------------\n");
		printf("Ban chon: ");
		scanf("%d",&chon);
		switch (chon)
		{
		case 0: printf("Dang thoat CT..."); break;
		case 1:
			int x;
			printf("Nhap gia tri nut: ");
			scanf("%d",&x);
			Push(plist, x); //gọi hàm thêm nút x vào đầu dslk
			printf("Danh sach lien ket don hien tai la:\n");
			Traverse(plist); //xuất giá trị các nút trong dslk
			break;
		case 2:
			if (Empty(plist))
				printf("D/s rong\n");
			else
			{
				int vtp;
				printf("Nhap vi tri nut p: ");
				scanf("%d", &vtp);
				//Xác định con trỏ của nút p (ở vị trí vtp) trong dslk
				NODEPTR p = NodePointer(plist, vtp);
				printf("Nhap gia tri nut: ");
				scanf("%d", &x);
				InsAfter(p,x);//Gọi hàm thêm nút x sau nút p trong dslk
				printf("Danh sach lien ket don hien tai la:\n");
				Traverse(plist); //xuất giá trị các nút trong dslk
			}
			break;
```

```
case 3:
	if (Empty(plist))
		printf("D/s rong\n");
	else
	{
		Pop(plist);
		printf("Danh sach lien ket don hien tai la:\n");
		Traverse(plist);//xuất giá trị các nút trong dslk
	}
	break;
case 4:
	if (Empty(plist))
		printf("D/s rong\n");
	else
	{
		int vtp;
		printf("Nhap vi tri nut p: ");
		scanf("%d", &vtp);
		//Xác định con trỏ của nút p (ở vị trí vtp) trong dslk
		NODEPTR p = NodePointer(plist, vtp);
		DelAfter(p);
		printf("Danh sach lien ket don hien tai la:\n");
		Traverse(plist);//xuất giá trị các nút trong dslk
	}
	break;
case 5:
	if (Empty(plist))
		printf("D/s rong\n");
	else
	{
		printf("Danh sach lien ket don hien tai la:\n");
		Traverse(plist);//xuất giá trị các nút trong dslk
	}
	break;
case 6:
	if (Empty(plist))
		printf("D/s rong\n");
	else
	{
		printf("Nhap gia tri nut: ");
		scanf("%d", &x);
		NODEPTR p = Search(plist, x);//Gọi hàm tìm nút x trong dslk
		if (p != NULL)
		{
			int posp = Position(plist, p);//gọi hàm xác định vị trí nút p
			printf("Tim thay tai vi tri thu %d\n", posp);
		}
		else
			printf("Khong tim thay.");
	}
	break;
```

```
		case 7:
			if (Empty(plist))
				printf("D/s rong\n");
			else
			{
				Sort(&plist);
				printf("Danh sach lien ket don hien tai la:\n");
				Traverse(plist);//xuất giá trị các nút trong dslk
			}
			break;
		case 8:
			printf("Nhap gia tri nut: ");
			scanf("%d", &x);
			if (Empty(plist))//nếu d/s rỗng
				Push(plist, x); //gọi hàm thêm nút x vào đầu dslk
			else
			{
				Sort(&plist);//gọi hàm sắp xếp d/s tăng dần theo trường info
				printf("Danh sach lien ket don hien tai la:\n");
				Traverse(plist);//xuất giá trị các nút trong dslk
				Place(&plist, x);//gọi hàm thêm nút x vào d/s đã được sắp thứ tự
			}
			printf("Danh sach lien ket don hien tai la:\n");
			Traverse(plist);//xuất giá trị các nút trong dslk
			break;
		case 9:
			if (Empty(plist))
				printf("D/s rong\n");
			else
			{
				ClearList(&plist);
				printf("D/s rong\n");
			}
			break;
		}
	} while (chon != 0);
	_getch();
}
```

(3) Bài tập về nhà:

Viết CT sử dụng dslk đơn để quản lý các SV với các chức năng sau:

a) Xem d/s sinh viên
b) Thêm sinh viên vào d/s
c) Xóa sinh viên trong d/s
d) Cập nhật thông tin sinh viên
e) Sắp xếp d/s sinh viên theo mã SV
f) Tìm kiếm sinh viên theo mã SV
g) Thêm sinh viên vào d/s đã có thứ tự theo mã SV
h) Xóa toàn bộ d/s sinh viên