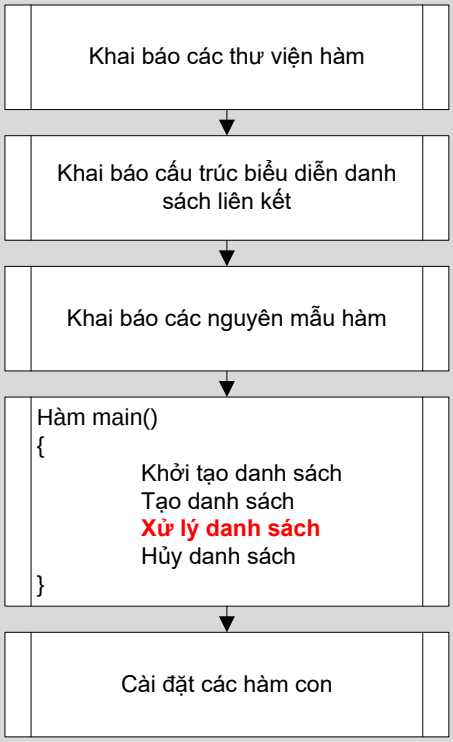
BÀI TẬP THỰC HÀNH SLL

Cấu trúc tổng quát của chương trình:



Chương trình mẫu:

- Quản lý danh sách liên kết đơn bằng 1 con trỏ HEAD

```
#include<stdio.h>
#include<stdlib.h>
int count=0;
struct node
{
	int data;
	struct node *next;
};

node *head,*newn,*p;
//-------------------------------------------------------------
void insert_at_begning(int value);
void insert_at_end(int value);
void create_list();
int display();
int main()
{
	char ch1;
	int value;
	head=NULL; //khoi tao danh sach rong
	do{
		create_list();
		display();
		fflush(stdin);
		printf("\ndo you want to create list ,y / n: ");
		scanf("%c",&ch1);
	}while(ch1=='y');

	printf("\nInsert at Head:");
	printf("\nEnter the value to be inserted:");
	scanf("%d",&value);
```

```
	insert_at_begning(value);
	display();

	printf("\nInsert at Tail:");
	printf("\nEnter value to be inserted:");
	scanf("%d",&value);
	insert_at_end(value);
	display();

}
//-----------------------------------------------------
void create_list()
{
	int value;
	struct node *temp;
	temp=head;
	newn=(struct node *)malloc(sizeof (struct node));
	printf("\nEnter the value to be inserted:");
	scanf("%d",&value);
	newn->data=value;
	if(head==NULL)
	{
		head=newn;
		head->next=NULL;
		count++;
	}
	else
	{
		insert_at_end(value);
		count++;
	}
}
//-----------------------------------------------------
void insert_at_begning(int value)
{
	newn=(struct node *)malloc(sizeof (struct node));
	newn->data=value;
	if(head==NULL)
	{
		head=newn;
		head->next=NULL;
		count++;
	}
	else
	{
		newn->next=head;
		head=newn;
		count++;
	}
}
//-------------------------------------------------------------
void insert_at_end(int value)
{
	struct node *temp;
	temp=head;
	newn=(struct node *)malloc(sizeof (struct node));
	newn->data=value;
	if(head==NULL)
	{
		head=newn;
		head->next=NULL;
		count++;
	}
	else
	{
		while(temp->next!=NULL)
			temp=temp->next;
		temp->next=newn;
```

```
            newn->next=NULL;
            count++;
      }
}
//-------------------------------------------------------
int display()
{
      p=head;
      while(p!=NULL)
      {
            printf("%d  ",p->data);
            p=p->next;
      }
}
//-------------------------------------------------------------------
```

- **Quản lý danh sách liên kết đơn bằng 2 con trỏ HEAD, TAIL**

```
#include <stdio.h>
#include <stdlib.h>
struct NODE
{
      int Data;
      struct NODE *pNext;
};

struct LIST
{
      NODE *pHead, *pTail;
};

void KhoiTao(LIST &L);
void Huy(LIST &L);
NODE *TaoNode(int x);
void ThemDau(LIST &L, NODE *p);
void Nhap(LIST &L);
void Xuat(LIST L);
int main()
{
      LIST L;
      Nhap(L);
      printf("\nDanh sach vua nhap:");
      Xuat(L);
      Huy(L);
}

void KhoiTao(LIST &L)
{
      L.pHead=L.pTail=NULL;
}

void Huy(LIST &L)
{
      NODE *p = (NODE*)malloc(sizeof(NODE));
      while(L.pHead!=NULL)
      {
            p=L.pHead;
            L.pHead=L.pHead->pNext;
            delete p;
      }
}

NODE *TaoNode(int x)
{
      NODE *p;
      p = (NODE*)malloc(sizeof(NODE));
      if(p==NULL)
      {
```

```
			printf("Khong cap phat duoc vung nho, ket thuc");
			exit(0);
		}
		p->Data=x;
		p->pNext=NULL;
		return p;
}
void ThemDau(LIST &L, NODE *p)
{
		if(L.pHead==NULL)
			L.pHead=L.pTail=p;
		else
		{
			p->pNext=L.pHead;
			L.pHead=p;
		}
}

void Nhap(LIST &L)
{
		int x;
		NODE *p;
		KhoiTao(L);
		do{
			printf("Nhap gia tri vao danh sach (Nhap 0 ket thuc):");
			scanf("%d",&x);
			if(x==0)
				break;
			p=TaoNode(x);
			ThemDau(L,p);
		}while(true);
}

void Xuat(LIST L)
{
		NODE *p=L.pHead;
		while(p!=NULL)
		{
			printf("%d  ",p->Data);
			p=p->pNext;
		}
}
```

BÀI TẬP: viết chương trình thực hiện các yêu cầu sau:

1. Thêm một phần tử vào đầu danh sách.
2. Thêm một phần tử vào cuối danh sách.
3. Xuất danh sách ra màn hình.
4. Liệt kê các phần tử mang giá trị chẵn.

```
void XuatChan(LIST l)
{
		NODE *p=l.pHead;
		while(p)
		{
			Nếu p->Data chẵn
			{
				in giá trị p->Data
			}
			p=p->pNext;
		}
}
```

5. Tính tổng các phần tử mang giá trị chẵn.

```
int TongChan(LIST l)
{
		NODE *p=l.pHead;
		int S=0;
		while(p!=NULL)
```

```
	{
		Nếu p->Data là số chẵn
			S=S+ p->Data;
		p=p->pNext;
	}
	return S;
}
```

6. Tìm phần tử có giá trị lớn nhất.

```
NODE *TimMax(LIST l)
{
	NODE *max=l.pHead;
	for(NODE *p=l.pHead->pNext; p; p=p->pNext)
	{
		Nếu giá trị của max < giá trị của p thì
		{
			gán lại max = p
		}
	}
	return max;
}
```

7. Đếm số lượng số nguyên tố trong danh sách.

bool LaSNT(int x); //Kiểm tra x có phải là số nguyên tố

int DemSNT(LIST l);//Đếm số lượng số nguyên tố trong danh sách

8. Thêm phần tử có giá trị nguyên X vào trước phần tử có giá trị chẵn đầu tiên trong danh sách. Nếu không có phần tử chẵn thì thêm vào đầu danh sách.

NODE *TimChanDau(LIST l);//Tìm chẵn đầu trong danh sách

void ThemkTruocp(LIST &l, NODE *p, NODE *k);//Thêm k vào trước p

```
void ThemXTruocChanDau(LIST &l, int X)//Thêm X vào trước chẵn đầu
{
	NODE *k=TaoNode(X);//Phần tử cần thêm
	NODE *p=TimChanDau(l);//Node có giá trị chẵn đầu tiên
	if(p==NULL)
	{
		ThemDau(l, k);
	}
	else
	{
		ThemkTruocp(l,  p, k);
	}
}
```

Ví dụ cách sử dụng hàm ThemXTruocChanDau()

```
void main()
{
	LIST l;
	int x;
	Nhap(l);
	printf("Danh sach vua nhap: \n");
	Xuat(l);
	printf("\nNhap gia tri can them vao truoc chan dau: ");
	scanf("%d",&x);
	ThemXTruocChanDau(l, x);
	printf("\nDanh sach sau khi them vao truoc chan dau:\n");
	Xuat(l);
}
```

9. Thêm phần tử có giá trị nguyên X vào sau phần tử có giá trị lẻ cuối cùng trong danh sách. Nếu không có phần tử lẽ thì thêm vào cuối danh sách.

NODE *TimLeCuoi(LIST l);//Tìm lẻ cuối cùng trong danh sách

void ThemCuoi(LIST &l, NODE *p);//Thêm p vào cuối danh sách

void ThemkSaup(LIST &l, NODE *p, NODE *k);//Thêm k vào sau p

void ThemXSauLeCuoi(LIST &l, int X);//Thêm X vào sau lẻ cuối

10. Xóa phần tử nhỏ nhất trong danh sách (Nếu trùng chỉ xóa phần tử nhỏ nhất đầu tiên).

    NODE *TimMin(LIST l);//Tìm node có giá trị nhỏ nhất

    void XoaDau(LIST &l);//Xóa node đầu của danh sách

    void XoaCuoi(LIST &l);//Xóa node cuối của danh sách

    void Xoap(LIST &l, NODE *p);//Xóa node p

    void XoaMin(LIST &l);//Xóa phần tử nhỏ nhất trong danh sách

11. Nhập vào phần tử X, xóa phần tử đứng sau và đứng trước phần tử X trong danh sách.

    NODE *TimX(LIST l, int X);//Tìm X

    void XoakTruocp(LIST &l, NODE *p, NODE *k);//Xóa k trước p

    void XoakSaup(LIST &l, NODE *p, NODE *q);//Xóa k sau p

12. Tách danh sách thành 2 danh sách, sao cho:
    - Danh sách thứ nhất chứa các phần tử là số nguyên tố.
    - Danh sách thứ hai chứa các phần tử còn lại.

```
void Tach(LIST l, LIST &l1, LIST &l2)
{
	KhoiTao(l1);
	KhoiTao(l2);
	NODE *p=l.pHead, *pAdd;
	while(p)
	{
		int k = p->Data;
		pAdd=TaoNode(k);
		Nếu k là số nguyên tố thì
			ThemDau(l1, pAdd);
		Ngược lại
			ThemDau(l2, pAdd);

		p trỏ đến node kế tiếp
	}
}
```