STRUCT AND ALGORITHMS

1 - 1

Single List

Phần 1: Cơ sở lý thuyết

Xem lại slide bài giảng

Phần 2: Thực hành

Bài tập 01: Xây dựng cấu trúc liên kết đơn để lưu danh sách số nguyên dương nhập từ bàn phím.
Hướng dẫn:

- Xây dựng cấu trúc danh sách liên kết đơn để có thể lưu được các số nguyên dương
- Viết code để cho phép nhập số nguyên dương. Nhập liên tiếp đến khi nhập -1 thì dừng quá trình trình

Xây dựng cấu trúc dữ liệu danh sách liên kết đơn

```
//Khai báo và khởi tạo danh sách liên kết đơn
#include <iostream>
using namespace std;

// khai báo cấu trúc danh sách liên kết đơn
typedef struct node
{
      int data;//Thành phần dữ liệu
      node* link;//Thành phần liên kết
}NODE;
typedef struct list
{
      NODE* first;//Con trỏ đầu danh sách
      NODE* last;//Con trỏ cuối danh sách
}LIST;

//Khởi tạo danh sách
void Init(LIST &list)
{
```

```
      list.first = list.last = NULL;//Khởi tạo danh sách rỗng
}
```

```
//Xây dựng cấu trúc Node cho danh sách
//Phương thức tạo động một cấu trúc node
  NODE* GetNode(int data)
{
      NODE *p;
      p = (NODE*)malloc(sizeof(NODE));
      //p = new NODE;
      if (p == NULL)
      {
            exit(1);
      }
      p->data = data;
      p->link = NULL;
      return p;
}
//Thêm node vào đầu danh sach
void AddFirst(LIST &list, NODE *new_node)
{
      if (list.first == NULL)
      {
            list.first = new_node;
            list.last = list.first;
      }
      else
      {
            new_node->link = list.first;
            list.first = new_node;
      }
}
//Phương thức thêm một node vào cuối danh sách
void AddLast(LIST &list, NODE *new_node)
{
      if (list.first == NULL)//dsanh rỗng
      {
            list.first = new_node;
            list.last = list.first;
      }
      else//danh sach khác rỗng
      {
            list.last->link = new_node;
            list.last = new_node;
      }
}
//Chú ý: Khi thực hiện xây dựng cấu trúc danh sách liên kết đơn,
```

tùy theo mục đích và yêu cầu xây dựng, người sử dụng có thể thực hiện 1 trong 2 thao tác thêm vào đầu hoặc thêm vào cuối danh sách. Vì đây là hướng dẫn sử dụng danh sách đơn nên Tôi xây dựng cả 2 cách và gọi xử lý chung.

```
//Việc thêm dữ liệu số vào trong danh sách liên kết đơn
//tham số Option để chỉ định cách thêm vào đầu hay vào cuối danh
sách.
void InsertList(LIST &list, int data,int option)
{
	NODE *p;
	p = GetNode(data);
	if (p != NULL)
	{
		if (option == 1)
		{
			AddLast(list, p);
		}
		else
		{
			AddFirst(list, p);
		}
	}
}
//Phương thức cho phép nhập dữ liệu số liên tục cho đến khi nhập -1
//tham số option để sử dụng cho trường hợp lựa chọn nhập tay hay
sinh số ngẫu nhiên.
Nếu là trường hợp sinh số ngẫu nhiên phải thêm khai báo #include
<ctime> cho việc sinh số ngẫu nhiên ở đầu chương trình. Với
option=1 là nhập tay.bằng tay
void CreateList(LIST &list,int option){
	if (option == 1){
		Init(list);
		int data;
		while (true){
			cout << "Nhap data: "; cin >> data;
			if (data != -1){
				InsertList(list, data, 0);//1. thêm vào đuôi
			}
			else{break;}}
	}
	else{
		//thêm bằng random
		Init(list);
		int n;
		cout << "so phan tu mong muon: "; cin >> n;
		int data;
```

```
            srand(time(0));
            for (int i = 0; i < n; i++){
                  data = 1 + rand() % (100 - 1 + 1);
//rand();//a+rand()%(b-a+1) hàm lấy giá trị ngẫu nhiên trong khoản;
                  InsertList(list, data, 0);//1. thêm vào đuôi
            }
      }

}
```

```
//Hiển thị dữ liệu
//Sử dụng vòng lặp để có thể duyệt từng phần tử trong danh sách.
void PrintList(LIST list)
{
      NODE *p = list.first;
      while (p!=NULL)
      {
            cout << p->data<<" ";
            p = p->link;
      }
}
//Hàm main để gọi chương trình
void main()
{
      LIST l;
      CreateList(l,1);
      PrintList(l);
      system("Pause");
}
```

Bài 02: Thực hiện một số thao tác với danh sách liên kết đơn

- Duyệt danh sách
    - In toàn bộ danh sách.
    - In danh sách các số chẵn, số lẻ, số nguyên tố (theo yêu cầu)
    - Tìm kiếm giá trị x nhập từ bàn phím trong danh sách.
    - Tìm kiếm tất cả các trí trị x có trong danh sách. có bao nhiêu giá trị x trong dsl
- Xóa danh sách
    - xóa giá trị x khỏi danh sách
    - xóa tất cả các giá trị x khỏi danh sách.
    - Xóa phần tử đầu danh sách.
    - xóa toàn bộ danh sách.

Hướng dẫn:

- Thực hiện thao tác xây dựng cấu trúc danh sách liên kết đơn giống bài 01
- Sau đó viết thêm các chức năng như trong nội dung yêu cầu

```
//hàm search tìm kiếm giá trị trong danh sách.
NODE* Search(LIST list, int x)
{
	NODE*p = list.first;
	while (p != NULL && p->data != x)
	{
		p = p->link;
	}
	return p;
}

//Thêm vào một node có giá trị data sau nút có giá trị x , x và
data nhập từ bàn phím
void AddAfterXValude(LIST &list,  int x,int data)
{
	//tìm ra được node có giá trị 2
	NODE* q = Search(list,x);
	NODE *new_node = GetNode(data);
	if (q != NULL&&new_node != NULL)
	{
		new_node->link = q->link;
		q->link = new_node;
		if (q == list.last)
			list.last = new_node;
	}
	else
	{
		cout << "khong ton tai node Q";
	}
	// thực hiện thao tác thêm vào sau node so 2
}
void RemoveFirst(LIST &list)
{
	if (list.first != NULL)
	{
		NODE*p = list.first;
		list.first = p->link;
		if (list.first == NULL)
			list.last = NULL;
		free(p);
		//delete p;
	}
}
void RemoveALL(LIST &l)
{
	if (l.first != NULL)
	{
```

```
            NODE*p = l.first;
            while (p != NULL)
            {
                  p = p->link;
                  RemoveFirst(l);
            }
      }
      else
      {
            cout << "danh sach da rong";
      }
}
```

```
void ThemVaoNodeSauQ(LIST &list, NODE *q, NODE* new_node)
{
      if (q != NULL&&new_node != NULL)
      {
            new_node->link = q->link;
            q->link = new_node;
            if (q == list.last)
                  list.last = new_node;
      }
}
NODE* Search(LIST list, int x)
{
      NODE* p = list.first;
      while (p != NULL)
      {
            if (p->data == x)
                  return p;
            else if( p==NULL)
            {
                  return p;
            }
            p = p->link;
      }
}
void RemoveFirst(LIST &list)
{
      if (list.first != NULL)
      {
            NODE*p = list.first;
            list.first = p->link;
            if (list.first == NULL)
            {
                  list.last = NULL;
            }
```

```
            free(p);
            //delete p;
      }
}
void RemoveNodeAfterQ(LIST &list, NODE*q)
{
      if (q != NULL && q->link != NULL)
      {
            NODE* p = q->link;
            q->link = p->link;
            if (p == list.last)
            {
                  list.last = q;
            }
            free(p);
      }
}
NODE* SearchNew(LIST list, int x)
{
      NODE* p = list.first;
      while (p != NULL&&p->data!=x)
      {
            p = p->link;
      }
}
void NhapDanhSach(LIST &list,int isFirst)
{
      int x = 0;
      while (x != -1)
      {
            cout << "\nNhap gia tri: "; cin >> x;
            if (x != -1)
            {
                  InsertList(list, x, isFirst);
            }
            cout << " Nhap -1 de thoat ra";
      }
}
void PrintList(LIST list)
{
      NODE* p;
      p = list.first;
      while (p!=NULL)
      {
            //1. thực hiện thao tác gì đó
            cout << p->data<<" ";
            //2. thành phần chạy
```

```
            p = p->link;
      }

}
int LaSoNguyenTo(int n)
{
      if (n < 2)
            return 0;

      for (int i = 2; i <= sqrt((float)n); i++)
      {
            if (n%i == 0)
            {
                  return 0;
            }
      }
      return 1;
}
int DemSoNguyenTo(LIST list)
{
      NODE* p;
      p = list.first;
      int dem = 0;
      while (p != NULL)
      {
            //1. thực hiện thao tác gì đó
            if (LaSoNguyenTo(p->data) == 1)
                  dem++;
            //2. thành phần chạy
            p = p->link;
      }
      return dem;
}
```

Các bài tập khác về danh sách liên kết đơn

Bài 03: Cho đa thức P(x)= anXn + an-1Xn-1 +... + a1X + a0. Hãy định nghĩa và xây dựng DSLK đơn lưu trữ đa thức P(x). Cài đặt các thao tác trên danh sách liên kết đơn biểu diễn đa thức:

- Lưu trữ đa thức.(Chú ý cách lưu trữ đảm bảo thứ tự giảm dần theo số mũ của từng hạng tử của đa thức).
  Ví dụ: Đa thức 5x4 - x + 3 được lưu trữ trong danh sách có 3 phần tử như sau:
- In đa thức
- Tính đạo hàm của P(x).

Hướng dẫn:

Bài 04: Định nghĩa và xây dựng DSLK đơn lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Lập trình đếm số node của danh sách chứa số nguyên tố. Xuất kết quả ra màn hình.

Hướng dẫn:

Bài 05: Định nghĩa và xây dựng DSLK đơn lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Lập trình hủy tất cả node chứa số nguyên tố ra khỏi danh sách. Xuất DSLK sau khi hủy tất cả node chứa số nguyên tố ra màn hình.

Bài 06: Định nghĩa và xây dựng DSLK đơn lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Lập trình đảo ngược thứ tự các phần tử của danh sách. Xuất DSLK đảo ngược ra màn hình.

Bài 07: Định nghĩa và xây dựng DSLK đơn lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Lập trình hủy node ở vị trí thứ k ra khỏi danh sách. Xuất DSLK sau khi hủy node ra màn hình.