STRUCT AND ALGORITHMS

1 - 2

Double List

Phần 1: Cơ sở lý thuyết

Xem lại slide bài giảng

Phần 2: Thực hành

- Bài 01: Xây dựng cấu trúc danh sách liên kết đôi để lưu trữ các số nguyên dương và thực hiện các thao tác với danh sách đó.

Hướng dẫn:
Xây dựng cấu trúc danh sách liên kết đôi

```
#include <iostream>
using namespace std;

typedef struct dnode
{
	int data;
	dnode* pre;
	dnode* next;
}DNODE;
typedef struct dlist
{
	DNODE* Head;
	DNODE* Tail;
}DLIST;
DNODE* GetDNode(int data)
{
	DNODE *new_Dnode;// = (DNODE*)malloc(sizeof(DNODE));
	new_Dnode = new DNODE;
	if (new_Dnode == NULL)
	{
		exit(0);
	}
	new_Dnode->data = data;
```

```
	new_Dnode->next = NULL;
	new_Dnode->pre = NULL;//# voi dslk đơn
	return new_Dnode;
}
void AddFirst(DLIST &dlist, DNODE* new_dnode)
{
	if (dlist.Head == NULL)
	{
		dlist.Head = new_dnode;
		dlist.Tail = dlist.Head;
	}
	else//dlist khác null
	{
		new_dnode->next = dlist.Head;//1
		dlist.Head->pre = new_dnode;//2 thêm trong danh sach đôi
		dlist.Head = new_dnode;//3
	}
}
void AddLast(DLIST &dlist, DNODE* new_Dnode)
{
	if (dlist.Head == NULL)
	{
		dlist.Head = new_Dnode;
		dlist.Tail = dlist.Head;
	}
	else
	{
		dlist.Tail->next = new_Dnode;//1
		new_Dnode->pre = dlist.Tail;//2
		dlist.Tail = new_Dnode;//3
	}
}
void InsertDlist(DLIST &dlist, int data,bool isFirst)
{
	DNODE*new_Dnode = GetDNode(data);
	if (new_Dnode != NULL)
	{
		if (isFirst==true)
		{
			AddFirst(dlist, new_Dnode);
		}
		else
		{
			AddLast(dlist, new_Dnode);
		}

	}
```

```
}
void Init(DLIST & dlist)
{
	dlist.Head = dlist.Tail = NULL;
}
void PrintDlist(DLIST dlist,bool isFirst)
{
	DNODE *p;
	if (isFirst==true)
	{
		p = dlist.Tail;
		cout << endl;
		while (p != NULL)
		{
			cout << p->data << " ";
			p = p->pre;
		}
	}
	else
	{
		p = dlist.Head;
		cout << endl;
		while (p != NULL)
		{
			cout << p->data << " ";
			p = p->next;
		}
	}
}
//Phương thức tạo một danh sách liên kết đôi
//cho phép insert vào ds đến khi nhập -1.
//Cho in danh sách theo thứ tự nhập
void CreateDList(DLIST &dlist,bool isFirst)
{
	int data;
	do
	{
		cout << "nhap data: "; cin >> data;
		if (data != -1)
		{
			InsertDlist(dlist, data, isFirst);
		}
	} while (data != -1);
	PrintDlist(dlist, isFirst);
}
void AddAfterQ(DLIST &dlist, DNODE *q,DNODE*new_Dnode)
{
```

```
	if (q != NULL)
	{
		DNODE*p = q->next;

		new_Dnode->next = p;//1
		new_Dnode->pre = q;//2
		q->next = new_Dnode;//3
		if (q != dlist.Tail)
		{
			p->pre = new_Dnode;//4d
		}
		else
		{
			dlist.Tail = new_Dnode;
		}

	}
	else
	{
		AddLast(dlist, new_Dnode);
	}
}
void AddBeforQ(DLIST &dlist, DNODE *q, DNODE*new_Dnode)
{
	if (q != NULL)
	{
		DNODE*p = q->pre;

		new_Dnode->next = q;//1
		new_Dnode->pre = p;//2
		p->next = new_Dnode;//3
		if (q != dlist.Head)
		{
			q->pre = new_Dnode;//4d
		}
		else
		{
			dlist.Head = new_Dnode;
		}

	}
	else
	{
		AddFirst(dlist, new_Dnode);
	}
}
```

```
int RemoveHead(DLIST &dlist)
{
	DNODE *p;
	int x=NULL;
	if (dlist.Head != NULL)
	{
		p= dlist.Head;
		x = p->data;
		dlist.Head = p->next;
		delete p;
		if (dlist.Head == NULL)
			dlist.Tail = NULL;
		else
		{
			dlist.Head->pre = NULL;
		}
	}
	return x;
}
int RemoveTail(DLIST &dlist)
{
	DNODE *p;
	int x = NULL;
	if (dlist.Head != NULL)
	{
		p = dlist.Tail;
		x = p->data;
		dlist.Tail = p->pre;
		delete p;
		if (dlist.Tail == NULL)
			dlist.Head = NULL;
		else
		{
			dlist.Tail->next = NULL;
		}
	}
	return x;
}
DNODE* SearchXValude(DLIST dlist, int x)
{
	DNODE*q = dlist.Head;
	while (q!=NULL&&q->data!=x)
	{
		q = q->next;
	}
	return q;
}
```

```
void InsertAfterXValue(DLIST &dlist, int x,int data,bool isBefore)
{
	DNODE *new_dnode = GetDNode(data);
	DNODE*q = SearchXValude(dlist, x);
	if (new_dnode != NULL)
	{
		if (isBefore == true)
		{
			AddBeforQ(dlist, q, new_dnode);
		}
		else
		{
			AddAfterQ(dlist, q, new_dnode);
		}

	}
}
typedef  struct nguoi{

	char  Hoten[30];
	int   So_CMND;
}NGUOI;
typedef struct ThongTinNguoi
{
	NGUOI BanThan,Cha,Me;
}ID;
void main()
{
	DLIST list;
	Init(list);

	CreateDList(list, false);
	InsertAfterXValue(list, 4,20,true);
	PrintDlist(list, false);
	RemoveHead(list);
	RemoveTail(list);
	PrintDlist(list, false);
	system("pause");
}
```

Một ví dụ khác về danh sách liên kết đôi:
Có cấu trúc menu trong gọi hàm.

```
//1.Tại cấu trúc chương trình để gọi chức năng theo dạng menu
//2. Các thao tác để xử lý danh sách liên kết Đôi
```

```
//2.1

//Bài tập về nhà
// Là thêm tất cả các vị trí có ghi chú là làm bài tập về nhà.
// Kiểm tra và thông báo nếu chọn lại chức năng 1 sẽ cảnh báo người
dùng. và cho phép người dùng lựa chọn khởi tạo lại danh sách hay
không?.

#include <iostream>
using namespace std;
//hàm menu cho phép chọn chức năng và trả về giá trị chọn tương ứng
int menu()
{
	cout << "=============Select menu==============" << endl;
	cout << "1.Init List"<<endl;
	cout << "2.Add" << endl;
	cout << "3.Print" << endl;
	cout << "4.Delete" << endl;
	cout << "5.Count element in List" << endl;
	cout << "6.Clear Display" << endl;
	cout << "7.Exit" << endl;
	cout << "=============Begin==============" << endl;
	int option;
	cout << "Select Function: "; cin >> option;
	return option;
}
//2. Init List
//khai báo cấu trúc danh sách liên kết đôi.
typedef struct dnode
{
	int data;
	dnode* pre;
	dnode* next;
}DNODE;
typedef struct dlist
{
	DNODE* head;
	DNODE* tail;

}DLIST;
void Init(DLIST &dlist)
{
	dlist.head = dlist.tail = NULL;
}
DNODE* GetNode(int data)
{
	DNODE * new_node = new DNODE;
```

```
	if (new_node == NULL)
	{
		exit(0);
	}
	new_node->data = data;
	new_node->pre = NULL;
	new_node->next = NULL;
	return new_node;
}
void AddFirst(DLIST &dlist, DNODE * new_node)
{
	if (dlist.head == NULL)//trường hợp danh sách rỗng
	{
		dlist.head = new_node;
		dlist.tail = dlist.head;
	}
	else// danh sách khác rỗng
	{
		new_node->next = dlist.head;
		dlist.head->pre = new_node;
		dlist.head = new_node;
	}
}
void InsertDList(DLIST & dlist, int data,int option)
{
	DNODE * new_node = GetNode(data);
	if (new_node != NULL)
	{
		switch (option)
		{
		case 1://addfirst
			AddFirst(dlist, new_node);
			break;
		case 2://addlast bài tập về nhà
			break;
		case 3://AfterQ bài tập về nhà
			break;
		case 4://beforeQ bài tập về nhà
			break;
		}

	}
}
void PrintList(DLIST dlist)
{
	DNODE *p = dlist.head;
	cout << "Current List: ";
```

```
	while (p!=NULL)
	{
		cout << p->data << " ";
		p = p->next;
	}
	cout << endl;
}
void DeleteHeadList(DLIST &dlist)
{
	if (dlist.head != NULL)
	{
		DNODE * p = dlist.head;//1
		dlist.head = p->next;//2
		if (p == dlist.tail)
		{
			dlist.tail = NULL;//3
		}
		else
		{
			dlist.head->pre = NULL;//3
		}
		delete p;//4
	}
}
void DeleteTailList(DLIST &dlist)
{
	if (dlist.head != NULL)
	{
		DNODE * p = dlist.tail;//1
		dlist.tail = p->pre;//2
		if (p == dlist.head)
		{
			dlist.head = NULL;//3
		}
		else
		{
			dlist.tail->next = NULL;//3
		}
		delete p;//4
	}
}
void DeleteAfterQ(DLIST &dlist, DNODE *q)
{
	if (q != NULL)
	{
		DNODE *p = q->next;
		if (p != NULL)
```

```
            {
                  q->next = p->next;
                  if (p == dlist.tail)
                  {
                        dlist.tail = q;
                  }
                  else
                  {
                        p->next->pre = q;
                  }
                  delete p;
            }

      }
}
void DeleteBeforeQ(DLIST &dlist, DNODE *q)
{
      if (q != NULL)
      {
            DNODE *p = q->next;
            if (p != NULL)
            {
                  q->next = p->next;
                  if (p == dlist.tail)
                  {
                        dlist.tail = q;
                  }
                  else
                  {
                        p->next->pre = q;
                  }
                  delete p;
            }

      }
}
DNODE* Search(DLIST dlidt, int k)
{
      DNODE *p = dlidt.head;
      while (p!=NULL&&p->data!=k)
      {
            p = p->next;
      }
      return p;
}
void DeleteValueK(DLIST &dlist, int k)
{
```

```
	//tìm k
	DNODE * p = Search(dlist, k);
	DNODE *q = NULL;
	//xóa k
	if (p != NULL)
	{
		if (p == dlist.head&&p == dlist.tail)
		{
			DeleteHeadList(dlist);
		}
		else if (p == dlist.head)
		{
			q = p->next;
			DeleteBeforeQ(dlist, q);
		}
		else
		{
			q = p->pre;
			DeleteAfterQ(dlist, q);
		}
	}
	else
	{
		cout << "Not exists " << k << "value into List";
	}
}
void main()
{
	int option=0;
	bool selected = false;//ghi nhận trạng thái chọn chức năng 1,
false (chưa chọn)
	DLIST dlist;
	do
	{
		option = menu();
		//các code gọi theo từng chức năng.
		switch (option)
		{
		case 1://khởi tạo danh sách
			system("cls");
			cout << "Khoi tao danh sach rong" << endl;
			Init(dlist);
			cout << "Successfull" << endl;
			selected = true;
			break;
		case 2:
			if (selected == true)
```

```
                {
                        system("cls");
                        cout << "Add new node into LIST" << endl;
                        int data;
                        cout << "Input value: "; cin >> data;
                        InsertDLList(dlist, data, 1);
                }
                else
                {
                        cout << "Not select 1 function";
                }
                break;
        case 3://khởi tạo danh sách
                if (selected == true)
                {
                system("cls");
                PrintList(dlist);
                }
                else
                {
                        cout << "Not select 1 function";
                }
                break;
        case 4://Delete list
                if (selected == true)
                {
                        system("cls");
                        DeleteHeadList(dlist);
                }
                else
                {
                        cout << "Not select 1 function";
                }
                break;
        case 6://khởi tạo danh sách
                system("cls");
                break;
        default:
                system("cls");
                cout << "Chua chon chuc nang" << endl;
                break;
        }

    } while (option>=1&&option<7);
}
```

Một số bài tập khác:

Bài 01: Định nghĩa và xây dựng DSLK đôi lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Lập trình đếm số node chứa số lẻ trong danh sách.

Bài 02: Định nghĩa và xây dựng DSLK đôi lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím.Chương trình có các chức năng:

- Nhập N số nguyên dương từ bàn phím.
- Tách DSLK đôi ban đầu thành hai DSLK đôi chẵn, DSLK đôi lẻ. Xuất ra màn hình hai DSLK đôi chẵn và DSLK đôi lẻ. (Với DSLK đôi chẵn chứa các số nguyên chẵn, DSLK đôi lẻ chứa các số nguyên lẻ)

Bài 03: Định nghĩa và xây dựng DSLK đôi lưu trữ dãy gồm N ≤ 1000 số thực được nhập từ bàn phím. Yêu cầu người dùng nhập số nguyên 0 ≤ k < N, hãy hủy số thực ở vị trí thứ k ra khỏi danh sách. Xuất DSLK sau khi hủy ra màn hình.

Bài 04: Định nghĩa và xây dựng DSLK đôi lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Yêu cầu người dùng nhập 2 số nguyên dương X và k, hãy tạo node chứa số nguyên X và chèn vào vị trí k của danh sách. Xuất DSLK sau khi chèn ra màn hình.

Bài 05: Định nghĩa và xây dựng DSLK đôi lưu trữ dãy gồm N ≤ 1000 số nguyên dương được nhập từ bàn phím. Hãy đếm số lượng node chứa giá trị lớn hơn giá trị của các node liền kề. Xuất kết quả ra màn hình.