Chương 7
CẤU TRÚC DỮ LIỆU
NGĂN XẾP-STACK

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

BÀI TẬP

- **Cho một ngăn xếp s và một đoạn chương trình sau:**

```
11. struct STACK s;
12. int x, y = 5;
13. Push(s, 8);
14. Push(s, y);
15. Push(s, 9);
16. Pop(s, x);
17. Push(s, 18);
18. Pop(s, x);
19. Push(s, 22);
20. while(IsEmpty(s) == 0)
21. {
22.     Pop(s, x);
23.     printf("%d ", y);
24. }
```

- **Hãy cho biết kết quả in ra màn hình khi thi hành đoạn chương trình trên là gì?**

BÀI TẬP

Bài làm

11. `struct STACK s;`
- Ngăn xếp rỗng

12. `int x, y = 5;`
- Giá trị biến y=5, x ko xác định

13. `Push(s, 8);`
- Ngăn xếp chứa (8)

14. `Push(s, y);`
- Ngăn xếp chứa (5,8)

15. `Push(s, 9);`
- Ngăn xếp chứa (9,5,8)

16. `Pop(s, x);`
- Ngăn xếp chứa (5,8), x=9

17. `Push(s, 18);`
- Ngăn xếp chứa (18,5,8)

18. `Pop(s, x);`
- Ngăn xếp chứa (5,8), x=18

19. `Push(s, 22);`
- Ngăn xếp chứa (22,5,8)

BÀI TẬP

```
20. while (IsEmpty(s) == 0)
21. {
22.     Pop(s, x);
23.     printf("%d ", y);
24. }
```

- **Lần lặp 1**
  1. Pop(s, x) x=22
  2. Ngăn xếp chứa (5,8)
  3. Xuất 5
- **Lần lặp 2**
  1. Pop(s, x) x=5
  2. Ngăn xếp chứa (8)
  3. Xuất 5
- **Lần lặp 3**
  1. Pop(s, x) x=8
  2. Ngăn xếp chứa ()
  3. Xuất 5
- **Kết luận: Đoạn chương trên xuất 5 5 5.**

BÀI TẬP

- Cho trước một cấu trúc **Stack S**.
- Cho trước các hàm thao tác trên Stack gồm:
  - **IsEmty**: Kiểm tra stack S có rỗng không? [1.Rỗng, 0. Không rỗng].
  - **IsFull**: Kiểm tra stack S có tràn không? [1.Tràn, 0. Không tràn].
  - **Push**: Thêm một phần tử vào Stack S.
  - **Pop**: lấy một phần tử ở đỉnh Stack S.
- Cho trước một cây nhị phân có nút gốc là Root.
- **Hãy viết hàm đếm số lượng nút trong cây mà không dùng giải thuật đệ quy**.

BÀI TẬP

- Cấu trúc dữ liệu:

```
11. struct node
12. {
13.     float info;
14.     struct node *pLeft;
15.     struct node *pRight;
16. };
17. typedef struct node NODE;
18. typedef NODE*TREE;
19. struct stack
20. {
21.     int n;
22.     NODE* a[100];
23. };
24. typedef struct stack STACK;
```

BÀI TẬP

- Định nghĩa hàm

```
11. void Init(STACK &st)
12. {
13.         st.n=0;
14. }
15. int IsEmpty(STACK st)
16. {
17.     if (st.n==0)
18.         return 1;
19.     return 0;
20. }
```

BÀI TẬP

```
11. int IsFulll(STACK st)
12. {
13.     if(st.n==100)
14.         return 1;
15.     return 0;
16. }
17. void Push(STACK&st,NODE* x)
18. {
19.     st.a[st.n]=x;
20.     st.n++;
21. }
22. NODE* Pop(STACK &st)
23. {
24.    NODE* x = st.a[st.n-1];
25.    st.n--;
26.    return x;
27. }
```

BÀI TẬP

```
11. int DemNode(TREE Root)
12. {
13.     int dem=0;
14.     STACK stk;
15.     init(stk);
16.     if (Root!=NULL)
17.         push(stk,Root);
18.     while(IsEmty(stk)==0)
19.     {
20.         NODE*p = pop(stk);
21.         dem++;
22.         if (p->pLeft)
23.             push(stk,p->pLeft);
24.         if (p->pRight)
25.             push(stk,p->pRight);
26.     }
27.     return dem;
28. }
```

BÀI TẬP

- **Bài 39: Anh Tri Tuấn**
- Cho một mảng hai chiều kích thước N×N. Phát sinh ngẫu nhiên giá trị các phần tử của mảng trong đoạn [0,5]. Viết chương trình nhập vào vị trí (i,j) bất kì, đếm số lượng các phần tử có cùng giá trị và liên thông với phần tử tại vị trí (i,j).
- **Thực hiện cùng yêu cầu trên nhưng dùng stack khử đệ quy.**

BÀI TẬP

Ví dụ

	0	1	2	3	4	5	6	7
0	5	0	1	2	2	1	2	4
1	2	1	2	3	3	5	2	5
2	1	1	2	1	3	4	5	2
3	2	3	5	5	4	4	3	5
4	1	3	3	4	5	2	5	1
5	4	1	2	3	3	5	1	2
6	4	4	2	1	5	1	3	2
7	1	3	4	2	3	5	4	5

Giả sử i=4, j=4

BÀI TẬP

Ví dụ

	0	1	2	3	4	5	6	7
0	5	0	1	2	2	1	2	4
1	2	1	2	3	3	5	2	5
2	1	1	2	1	3	4	5	2
3	2	3	5	5	4	4	3	5
4	1	3	3	4	5	2	5	1
5	4	1	2	3	3	5	1	2
6	4	4	2	1	5	1	3	2
7	1	3	4	2	3	5	4	5

Giả sử i=4, j=4

BÀI TẬP

Đếm ô liên thông với i=4,j=4

	0	1	2	3	4	5	6	7
0	5	0	1	2	2	1	2	4
1	2	1	2	3	3	5	2	5
2	1	1	2	1	3	4	5	2
3	2	3	5	5	4	4	3	5
4	1	3	3	4	5	2	5	1
5	4	1	2	3	3	5	1	2
6	4	4	2	1	5	1	3	2
7	1	3	4	2	3	5	4	5

Số lượng ô liên thông là: 11 ô

BÀI TẬP

Lân Cận

(d-1,c-1)	(d-1,c+0)	(d-1,c+1)
(d+0,c-1)	(d,c)	(d+0,c+1)
(d+1,c-1)	(d+1,c+0)	(d+1,c+1)

Độ lệch

(-1,-1)	(-1,+0)	(-1,+1)
(+0,-1)	(d,c)	(+0,+1)
(+1,-1)	(+1,+0)	(+1,+1)

Khai báo

```
int di[8]= {-1, -1, -1, 0, +1, +1, +1, 0};
int dj[8]= {-1, 0, +1, +1, +1, 0, -1 -1};
```

BÀI TẬP

```
1. int DemLoang(int a[][100],
    int n,int d, int c,int gt)
2. {
3.    if(d<0||d>=n||c<0||c>=n)
4.       return 0;
5.    if(a[d][c]!=gt)
6.       return 0;
7.    int dem=1;
8.    a[d][c]=-1;
9.    int di[8]={-1,-1,-1,0,1,1,1,0};
10.   int dj[8]={-1,0,1,1,1,0,-1,-1};
11.   for(int k=0;k<8;k++)
12.      dem+=DemLoang(a,n,
            d+di[k],c+dj[k],gt);
13.   return dem;
14. }
```

BÀI TẬP

Đếm ô liên thông với i=4,j=4

	0	1	2	3	4	5	6	7
0	5	0	1	2	2	1	2	4
1	2	1	2	3	3	-1	2	-1
2	1	1	2	1	3	4	-1	2
3	2	3	-1	-1	4	4	3	-1
4	1	3	3	4	-1	2	-1	1
5	4	1	2	3	3	-1	1	2
6	4	4	2	1	-1	1	3	2
7	1	3	4	2	3	-1	4	5

Số lượng ô liên thông là: 11 ô

BÀI TẬP

```
11. int DemLanCan(int a[][100],
        int n,int i, int j)
12. {
13.     int gt=a[i][j];
14.     int dem=DemLoang(a,n,
                i,j,gt);
15.     for(int k=0;k<n;k++)
16.       for(int l=0;l<n;l++)
17.         if(a[k][l]==-1)
18.           a[k][l]=gt;
19.     return dem;
20. }
```

BÀI TẬP

– Cấu trúc dữ liệu

```
11. struct dt
12. {
13.     int d;
14.     int c;
15. };
16. typedef struct dt DT;
17. struct stack
18. {
19.     int n;
20.     DT a[100];
21. };
22. typedef struct stack STACK;
```

BÀI TẬP

```
11. void Init(STACK &st)
12. {
13.     st.n=0;
14. }
15. int IsEmpty(STACK st)
16. {
17.     if(st.n==0)
18.         return 1;
19.     return 0;
20. }
21. int IsFull(STACK st)
22. {
23.     if(st.n==100)
24.         return 1;
25.     return 0;
26. }
```

BÀI TẬP

```
11. void Push(STACK &st,DT x)
12. {
13.     st.a[st.n]=x;
14.     st.n++;
15. }
16. DT Pop(STACK &st)
17. {
18.     DT x=st.a[st.n-1];
19.     st.n--;
20.     return x;
21. }
```

```
11. int DemLanCan(int a[][100],
                  int n,int i, int j)
12. {
13.     int dem=0;
14.     int gt=a[i][j];
15.     STACK stk;
16.     Init(stk);
17.     DT ob={i,j};
18.     Push(stk,ob);
19.     int di[8]={-1,-1,-1,0,1,1,1,0};
20.     int dj[8]={-1,0,1,1,1,0,-1,-1};
21.     ...
22. }
```

```
11. int DemLanCan(int a[][100],
                  int n,int i, int j)
12. {
13.   ...
14.   while(IsEmpty(st)==0)
15.   {
16.     ob = Pop(stk);
17.     dem++;
18.     int d=ob.d;
19.     int c=ob.c;
20.     a[d][c]=-1;
21.     for(int k=0;k<8;k++)
22.       if(d+di[k]>=0&&d+di[k]<n &&
23.          c+dj[k]>=0&&c+dj[k]<n &&
24.          a[d+di[k]][c+dj[k]]==gt)
25.       {
26.         ob.d = d+di[k];
27.         ob.c = c+dj[k];
28.         Push(stk,ob);
29.       }
30.   }
31.   ...
32. }
```

```
1. int DemLanCan(int a[][100],
                 int n,int i, int j)
2. {
3.   ...
4.   for(int k=0;k<n;k++)
5.     for(int l=0;l<n;l++)
6.       if(a[k][l]==-1)
7.         a[k][l]=gt;
8.   return dem;
9. }
```

BÀI TẬP

- Hãy cài đặt thuật toán Quick Sort sắp xếp mảng một chiều các số thực bằng hai phương pháp.
  - Đệ quy
  - Không đệ quy bằng cách sử dụng kỹ thuật Stack.

BÀI TẬP LUYỆN TẬP

- Phương pháp đệ quy- Hàm cài đặt

```
11. void SapTang(float a[], int n)
12. {
13.     QuickSort(a,0,n-1);
14. }
15. void HoanVi(float&a,float&b)
16. {
17.     float temp = a;
18.     a = b;
19.     b = temp;
20. }
```

4. MỘT CÁCH CÀI ĐẶT KHÁC

```
11. void QuickSort(float a[],
                   int Left,int Right)
12. {
13.     if(Left<Right)
14.     {
15.         int m1,m2;
16.         Partition(a,Left,Right,
                      m1,m2);
17.         QuickSort(a,Left,m1);
18.         QuickSort(a,m2,Right);
19.     }
20. }
```

```
10. void Partition(int a[],
        int Left, int Right,
        int &m1,int &m2)
11. {
12.     int pivot=a[(Left+Right)/2];
13.     int low = Left;
14.     int high = Right;
15.     while(low<high)
16.     {
17.         while (a[low]<pivot)
18.             low++;
19.         while (a[high]>pivot)
20.             high--;
21.         if(low<=high)
22.         {
23.             HoanVi(a[low],a[high]);
24.             low++;
25.             high--;
26.         }
27.     }
28.     m1 = high;
29.     m2 = low;
30. }
```

BÀI TẬP

- **Cấu trúc dữ liệu**

```
11. struct doan
12. {
13.     int d;
14.     int c;
15. };
16. typedef struct doan DOAN;
17. struct stack
18. {
19.     int n;
20.     DOAN a[100];
21. };
22. typedef struct stack STACK;
```

BÀI TẬP

```
11. void Init(STACK &st)
12. {
13.     st.n=0;
14. }
15. int IsEmpty(STACK st)
16. {
17.     if(st.n==0)
18.         return 1;
19.     return 0;
20. }
21. int IsFull(STACK st)
22. {
23.     if(st.n==100)
24.         return 1;
25.     return 0;
26. }
```

BÀI TẬP

```
11. void Push(STACK &st,DOAN x)
12. {
13.     st.a[st.n]=x;
14.     st.n++;
15. }
16. DOAN Pop(STACK &st)
17. {
18.     DOAN x=st.a[st.n-1];
19.     st.n--;
20.     return x;
21. }
```

BÀI TẬP LUYỆN TẬP

- Hàm cài đặt

```
1. void HoanVi(float&a,float&b)
2. {
3.     float temp=a;
4.     a=b;
5.     b=temp;
6. }
```

```
10. void Partition(int a[],
        int Left, int Right,
        int &m1,int &m2)
11. {
12.     int pivot=a[(Left+Right)/2];
13.     int low = Left;
14.     int high = Right;
15.     while(low<high)
16.     {
17.         while (a[low]<pivot)
18.             low++;
19.         while (a[high]>pivot)
20.             high--;
21.         if(low<=high)
22.         {
23.             HoanVi(a[low],a[high]);
24.             low++;
25.             high--;
26.         }
27.     }
28.     m1 = high;
29.     m2 = low;
30. }
```

```
11. void QuickSort(float a[],int n)
12. {
13.     if(n<=1)
14.         return;
15.     DOAN ob={0,n-1};
16.     STACK stk;
17.     Init(stk);
18.     Push(stk,ob);
19.     …
20. }
```

```
11. void QuickSort(float a[],int n)
12. {
13.   …
14.   while(IsEmpty(stk)==0)
15.   {
16.     ob = Pop(stk);
17.     int m1,m2;
18.   Partition(a,ob.d,ob.c,m1,m2);
19.     if(ob.d<m1)
20.     {
21.       DOAN ob1={ob.d,m1};
22.       Push(stk,ob1);
23.     }
24.     if(m2<ob.c)
25.     {
26.       DOAN ob2={m2,ob.c};
27.       Push(stk,ob2);
28.     }
29.   }
30. }
```

```
11. void QuickSort(float a[],int n)
12. {
13.     if(n<=1)
14.         return;
15.     DOAN ob={0,n-1};
16.     STACK stk;
17.     Init(stk);
18.     Push(stk,ob);
19.     while(IsEmpty(stk)==0)
20.     {
21.         ob = Pop(stk);
22.         int m1,m2;
23.         Partition(a,ob.d,ob.c,m1,m2);
24.         if(ob.d<m1)
25.         {
26.             DOAN ob1={ob.d,m1};
27.             Push(stk,ob1);
28.         }
29.         if(m2<ob.c)
30.         {
31.             DOAN ob2={m2,ob.c};
32.             Push(stk,ob2);
33.         }
34.     }
35. }
```

BÀI TẬP

- **Bài 38: Anh Tri Tuấn**
- Bài dãy số f(n)=1 nếu n=0 hay n=1. Ngoài ra: f(n)=f(n-1)+f(n-2) nếu n>1.
- Viết hàm đệ quy tính f(n).
- Viết hàm không đệ quy tính f(n) băng cách sử dụng kỹ thuật Stack.

BÀI TẬP

- Viết hàm đệ quy tính f(n)

```
1. long fibo(int n)
2. {
3.       if(n==0 || n==1)
4.           return 1;
5.       return fibo(n-1)+
                 fibo(n-2);
6. }
```

BÀI TẬP

- Viết hàm khử đệ quy tính f(n)

```
1.  long fibo(int n)
2.  {
3.      if(n==0 || n==1)
4.          return 1;
5.      long ftt=1;
6.      long ft=1;
7.      long fhh;
8.      int i=2;
9.      while(i<=n)
10.     {
11.         fhh=ft+ftt;
12.         i++;
13.         ftt=ft;
14.         ft=fhh;
15.     }
16.     return fhh;
17. }
```

BÀI TẬP

- Hãy cài đặt hàm tính chiều cao của cây nhị phân các số nguyên bằng hai phương pháp.
  - Đệ quy
  - Không đệ quy bằng cách sử dụng kỹ thuật Stack.

BÀI TẬP

- Cấu trúc dữ liệu.

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```

BÀI TẬP

- Định nghĩa hàm

```
11. int ChieuCao(TREE t)
12. {
13.     if(!t)
14.         return 0;
15.     int a=ChieuCao(t->pLeft);
16.     int b=ChieuCao(t->pRight);
17.     if(a>b)
18.         return (a+1);
19.     return (b+1);
20. }
```

BÀI TẬP

- Cấu trúc dữ liệu.

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```

BÀI TẬP

- Cấu trúc dữ liệu.

```
11. struct dt
12. {
13.     int height;
14.     NODE *pNode;
15. };
16. typedef struct dt DT;
17. struct stack
18. {
19.     int n;
20.     DT a[100];
21. };
22. typedef struct stack STACK;
```

BÀI TẬP

```
11. void Init(STACK &st)
12. {
13.     st.n=0;
14. }
15. int IsEmpty(STACK st)
16. {
17.     if(st.n==0)
18.         return 1;
19.     return 0;
20. }
21. int IsFull(STACK st)
22. {
23.     if(st.n==100)
24.         return 1;
25.     return 0;
26. }
```

BÀI TẬP

```
11. void Push(STACK &st,DT x)
12. {
13.     st.a[st.n] = x;
14.     st.n++;
15. }
16. DT Pop(STACK &st)
17. {
18.     DT x = st.a[st.n-1];
19.     st.n--;
20.     return x;
21. }
```

```
11. int ChieuCao(TREE t)
12. {
13.     if(t==NULL)
14.         return 0;
15.     DT ob={1,t};
16.     STACK stk;
17.     Init(stk);
18.     Push(stk,ob);
19.     int lc = 0;
20.     …
21. }
```

```
11. int ChieuCao(TREE t)
12. {
13.   …
14.   while(IsEmpty(stk)==0)
15.   {
16.     ob = Pop(stk);
17.     if(ob.height>lc)
18.       lc  = ob.height;
19.     if(ob.pNode->pLeft)
20.     {
21.       DT ob1={ob.height+1,
                  ob.pNode->pLeft};
22.       Push(stk,ob1);
23.     }
24.     if(ob.pNode->pRight)
25.     {
26.       DT ob2={ob.height+1,
                  ob.pNode->pRight};
27.       Push(stk,ob2);
28.     }
29.   }
30.   return lc;
31. }
```