STRUCT AND ALGORITHMS

2 - 1

Stack & Queue

Phần 1: Cơ sở lý thuyết

Xem lại slide bài giảng

Phần 2: Thực hành

Cấu trúc dữ liệu Stack

- xây dựng cấu trúc Stack
  - cài đặt bằng mảng.
  - cài đặt bằng danh sách (làm file riêng)
- thực hiện các thao tác.
  - khởi tạo stack
  - kiểm tra stack rỗng, đầy
  - Thêm phần tử vào stack
  - Lấy phần tử đầu stack
  - Thực hiện input output

Bài tập 01: Xây dựng cấu trúc stack với các thao tác của Stack. Sau đó ứng dụng vào việc chuyển đổi cơ số của 1 số.

```
#include <iostream>
using namespace std;
int menu()
{
	int option;
	cout << "-----Menu----";
	cout << "\n1. Init Stack";
	cout << "\n2. Check Stack";
	cout << "\n3. Push into Stack";
	cout << "\n4. Pop out Stack";
```

```
	cout << "\n5. Get item top Stack";
	cout << "\n6. translate 10 to 2";
	cout << "\n7. Exit";
	cout << "seleted Option: "; cin >> option;
	return option;
}
//Khai báo stack bằng mảng
#define max 100
typedef int item;
struct Stack
{
	int count;
	int top;// đỉnh stack
	item data[max];//mảng chứa các phần tử của stack
};
void Init(Stack &stack)
{
	stack.top = 0;
	stack.count = 0;
}
int IsEmpty(Stack stack)
{
	if (stack.top == 0)
		return 1;
	return 0;
}
int isFull(Stack stack)
{
	if (stack.top == max)
		return 1;
	return 0;
}
void Push(Stack &stack, int x)
{
	if (isFull(stack) == 0)
	{
		stack.data[stack.top] = x;
		stack.top++;
		stack.count++;
	}
	else
	{
		cout << "Stack full";
	}
}
item Pop(Stack &stack)
{
```

```
	item data=NULL;
	if (!IsEmpty(stack))
	{
		stack.top--;
		data = stack.data[stack.top];
		stack.count--;
	}
	return data;
}


void DoiCoSo(int y)
{
	int mod;
	Stack s;
	if (y >= 0)
	{
		do
		{
			mod = y % 2;
			Push(s, mod);
			y = y / 2;

		} while (y != 0);
	}
	//lấy ra
	while (IsEmpty(s) == 0)
	{
		cout << Pop(s) << " ";
	}
}
void main()
{
	int option = 0;
	Stack stack;
	do
	{
		option = menu();
		switch (option)
		{
		case 1:
			system("clr");
			cout << "Init Stack";
			Init(stack);
			break;
		case 2:
			system("clr");
```

```
                cout << "Checked Stack";
                if (IsEmpty(stack) == 1)
                      cout << "Stack empty";
                else if (isFull(stack)==1)
                {
                      cout << "Stack Full";
                }
                else
                {
                      cout << "Stack have "<<stack.count<<" item";
                }
                break;
           }
     } while (option!=6);
}
```

Một ví dụ khác về Stack

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

int menu()
{
      cout << "=============Select menu==============" << endl;
      cout << "Moi ban chon chuc nang:";
      cout << "\n1: Kiem tra Stack rong";
      cout << "\n2: Kiem tra Stack day";
      cout << "\n3: Them phan tu vao Stack";
      cout << "\n4: Xoa phan tu trong Stack";
      cout << "\n5: Xuat Stack";
      cout << "\n6: Thoat";
      cout << "=============Begin==============" << endl;
      int option;
      cout << "Select Function: "; cin >> option;
      return option;
}
#define Max 100 //so phan tu toi da cua Stack
typedef int item; //kieu du lieu cua Stack
struct Stack
{
      int Top; //Dinh Top
      item Data[Max]; //Mang cac phan tu
};

void Init(Stack &S); //khoi tao Stack rong
int Isempty(Stack S); //kiem tra Stack rong
int Isfull(Stack S); //kiem tra Stack day
void Push(Stack &S, item x); //them phan tu vao Stack
```

```
int Peak(Stack S); //Lay phan tu o dau Stack nhung khong xoa
int Pop(Stack &S); //Loai bo phan tu khoi Stack
void Input(Stack &S); //Nhap Stack
void Output(Stack S); //Xuat Stack

void Init(Stack &S) //khoi tao Stack rong
{
	S.Top = 0; //Stack rong khi Top la 0
}

int Isempty(Stack S) //kiem tra Stack rong
{
	return (S.Top == 0);
}

int Isfull(Stack S) //kiem tra Stack day
{
	return (S.Top == Max); //
}

void Push(Stack &S, item x) //them phan tu vao Stack
{
	if (!Isfull(S))
	{
		S.Data[S.Top] = x; //Gan du lieu
		S.Top++; //Tang Top len 1
	}
}

int Peak(Stack S) //Lay phan tu o dau Stack nhung khong xoa
{
	return S.Data[S.Top - 1]; //Lay du lieu tai Top
}

int Pop(Stack &S) //Loai bo phan tu khoi Stack
{
	if (!Isempty(S))
	{
		S.Top--; //Giam Top
		return S.Data[S.Top]; //Lay du lieu tai Top
	}
}

void Input(Stack &S)
{
	int n;
	item x;
```

```
	do
	{
		printf("Nhap so phan tu cua Stack (<%d) :", Max);
		scanf("%d", &n);
	} while (n>Max || n<1);
	for (int i = 0; i<n; i++)
	{
		printf("Nhap phan tu thu %d : ", i + 1);
		scanf("%d", &x);
		Push(S, x);
	}
}

void Output(Stack S)
{
	for (int i = S.Top - 1; i >= 0; i--)
		printf("%d   ", S.Data[i]);
	printf("\n");
}
void main()
{
	int option = 0;
	bool selected = false;//ghi nhận trạng thái chọn chức năng 1,
false (chưa chọn)
	Stack S;
	Init(S);
	Input(S);
	Output(S);
	do
	{
		option = menu();
		//các code gọi theo từng chức năng.
		switch (option)
		{
		case 1://
			system("cls");
			if (Isempty(S)) printf("Stack rong !");
			else printf("Stack khong rong !");selected = true;
			break;
		case 2:
			if (selected == true)
			{
				if (Isfull(S)) printf("Stack day !");
				else printf("Stack chua day !");
			}
			else
			{
```

```
                    cout << "Not select 1 function";
               }
               break;
          case 3://khởi tạo danh sách
               if (selected == true)
               {
                    item x;
                    printf("Nhap phan tu can chen vao DS: ");
                    scanf("%d", &x);
                    Push(S, x);
               }
               else
               {
                    cout << "Not select 1 function";
               }
               break;
          case 4://Delete list
               if (selected == true)
               {
                    Pop(S);
               }
               else
               {
                    cout << "Not select 1 function";
               }
               break;
          case 5://khởi tạo danh sách
               system("cls");
               Output(S);
               break;
          default:
               system("cls");
               cout << "Chua chon chuc nang" << endl;
               break;
          }

     } while (option !=6);
}
```

Bài tập xây dựng cấu trúc stack bằng cấu trúc danh sách liên kết đơn

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

typedef int item; //kieu du lieu
struct Node
{
```

```
	item Data; //du lieu
	Node *Next; //link
};
typedef struct Stack
{
	Node *Top;
};

void Init(Stack &S); //khoi tao Stack rong
int Isempty(Stack S); //kiem tra Stack rong
int Len(Stack S); //Do dai Stack
void Push(Stack &S, item x); //them phan tu vao Stack
int Peak(Stack S); //Lay phan tu o dau Stack nhung khong xoa
int Pop(Stack &S); //Loai bo phan tu khoi Stack
void Input(Stack &S); //Nhap Stack
void Output(Stack S); //Xuat Stack
Node *MakeNode(item x); //Tao 1 Node

void Init(Stack &S) //khoi tao Stack rong
{
	S.Top = NULL;
}

int Isempty(Stack S) //kiem tra Stack rong
{
	return (S.Top == NULL);
}

int Len(Stack S)
{
	Node *P = S.Top;
	int i = 0;
	while (P != NULL) //trong khi chua het Stack thi van duyet
	{
		i++;
		P = P->Next;
	}
	return i;
}

Node *MakeNode(item x) //tao 1 Node
{
	Node *P = (Node*)malloc(sizeof(Node));
	P->Next = NULL;
	P->Data = x;
	return P;
}
```

```
void Push(Stack &S, item x) //them phan tu vao Stack
{
	Node *P = MakeNode(x);
	P->Next = S.Top;
	S.Top = P;
}

int Peak(Stack S) //Lay phan tu o dau Stack nhung khong xoa
{
	return S.Top->Data;
}

int Pop(Stack &S) //Loai bo phan tu khoi Stack
{
	if (!Isempty(S))
	{
		item x = S.Top->Data; //luu lai gia tri
		S.Top = S.Top->Next; //Xoa phan tu Top
		return x;
	}
}

void Input(Stack &S) //nhap danh sach
{
	int i = 0;
	item x;
	do
	{
		i++;
		cout<<"Nhap phan tu thu "<< i;
		scanf("%d", &x);
		if (x != 0) Push(S, x);
	} while (x != 0); //nhap 0 de ket thuc
}

void Output(Stack S)
{
	Node *P = S.Top;
	while (P != NULL)
	{
		printf("%d   ", P->Data);
		P = P->Next;
	}
	printf("\n");
}
```

```
int main()
{
	Stack S;
	Init(S);
	Input(S);
	Output(S);

	int lua_chon;
	cout<<"Moi ban chon phep toan voi DS LKD:";
	cout << "\n1: Kiem tra Stack rong";
	cout << "\n2: Do dai Stack";
	cout << "\n3: Them phan tu vao Stack";
	cout << "\n4: Xoa phan tu trong Stack";
	cout << "\n5: Xuat Stack";
	cout << "\n6: Thoat";
	do
	{
		cout << "\nBan chon: ";
		cin>>lua_chon;
		switch (lua_chon)
		{
		case 1:
		{
			if (Isempty(S))
				cout << "Stack rong !";
			else cout << "Stack khong rong !";
			break;
		}
		case 2:
		{
			cout << "Do dai Stack:"<< Len(S);
			break;
		}
		case 3:
		{
			item x;
			cout << "Nhap phan tu can chen vao DS: ";
			cin>>x;
			Push(S, x);
			break;
		}
		case 4:
		{
			Pop(S);
			break;
		}
		case 5:
```

```
            {
                  Output(S);
                  break;
            }
            case 6: break;
            }
      } while (lua_chon != 6);
      return 0;
}
```

Một số bài tập khác về Stack

1. Cho chuỗi biểu thức chỉ gồm các cặp dấu ngoặc ‘(‘, ‘)’, ‘[‘, ‘]’ hoặc ‘{‘, ‘}’. Hãy kiểm tra các dấu ngoặc được đóng mở có hợp lệ cho một biểu thức toán học hay không.
2. Viết chương trình định nghĩa và xây dựng ngăn xếp stack lưu trữ một xâu kí tự nhập vào từ bàn phím. Sau đó xác định xâu kí tự là palindrome hay không.
   Lưu ý: Xâu kí tự được gọi là palindrome nếu nó không thay đổi khi ta đảo ngược thứ tự của các kí tự trong xâu kí tự
   Ví dụ : ABBA, ABCBA, 45811854 …
3. Cho số nguyên dương n. Hãy ứng dụng cấu trúc Stack để chuyển đổi n sang hệ cơ số d (biết $2 \le d \le 16$).
4. Nhập $N \le 1000$ số nguyên từ bàn phím. Với mỗi số nhập vào, thêm vào Stack S nếu là số chẵn, ngược lại thêm vào Queue Q nếu là số lẻ. Xuất các phần tử có trong S và Q ra màn hình, dòng đầu tiên là các phần tử của S, dòng tiếp theo là các phần tử của Q.

Phần 2: Queue

Bài 01: Xây dựng cấu trúc Queue và các thao tác của cấu trúc này

- Khởi tạo Queue rỗng
- Kiểm tra Queue rỗng
- Kiểm tra Queue đầy
- Thêm phần tử vào cuối Queue
- Loại bỏ phần tử khỏi đầu Queue
- Xem thông tin phần tử đều Queue
- Thêm phần tử vào cuối Queue vòng.
- Loại bỏ phần tử ở đầu Queue vòng.

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

#define Max 5 //so phan tu toi da cua Queue
typedef int item; //kieu du lieu

struct Queue
{
	int Front, Rear; //front: phan tu dau hang, rear: phan tu cuoi
hang
	item Data[Max]; //Mang cac phan tu
	int count; //dem so phan tu cua Queue
};

void Init(Queue &Q); //khoi tao Queue rong
int Isempty(Queue Q); //kiem tra Queue rong
int Isfull(Queue Q); //kiem tra Queue day
void Push(Queue &Q, item x); //them phan tu vao cuoi hang doi
int Pop(Queue &Q); //Loai bo phan tu khoi dau hang doi
int Qfront(Queue Q); //xem thong tin phan tu dau hang doi
void Push_Circular(Queue &Q, item x); //them phan tu vao cuoi hang
doi vong
int Pop_Circular(Queue &Q); //Loai bo phan tu khoi dau hang doi
vong
void Input(Queue &Q); //Nhap
void Output(Queue Q); //Xuat

void Init(Queue &Q) //khoi tao Queue rong
{
	Q.Front = 0; //phan tu dau
	Q.Rear = -1; // phan tu cuoi o -1 (khong co phan tu trong Q)
```

```
	Q.count = 0; //so phan tu bang 0
}

int Isempty(Queue Q) //kiem tra Queue rong
{
	if (Q.count == 0) //so phan tu = 0 => rong
		return 1;
	return 0;
}

int Isfull(Queue Q) //kiem tra Queue day
{
	if (Q.count == Max) //so phan tu = Max => day
		return 1;
	return 0;
}

void Push(Queue &Q, item x) //them phan tu vao cuoi Queue
{
	if (Isfull(Q)) cout<<"Hang doi day !";
	else
	{
		Q.Data[++Q.Rear] = x; //tang Rear len va gan phan tu vao
		Q.count++; //tang so phan tu len
	}
}

int Pop(Queue &Q) //Loai bo phan tu khoi dau hang doi
{
	if (Isempty(Q)) cout << "Hang doi rong !";
	else
	{
		item x = Q.Data[Q.Front];
		for (int i = Q.Front; i<Q.Rear; i++) //di chuyen cac phan
tu ve dau hang
			Q.Data[i] = Q.Data[i + 1];
		Q.Rear--; // giam vi tri phan tu cuoi xuong
		Q.count--;//giam so phan tu xuong
		return x; //tra ve phan tu lay ra
	}
}

item Qfront(Queue Q) //xem thong tin phan tu dau hang
{
	if (Isempty(Q)) cout << "Hang doi rong !";
	else return Q.Data[Q.Front];
}
```

```
void Push_Circular(Queue &Q, item x) //them phan tu vao cuoi hang
doi vong
{
      if (Isfull(Q)) cout << "Hang doi day !";
      else
      {
            Q.Data[(++Q.Rear) % Max] = x;
            //tang Rear len va gan phan tu vao, Neu Rear dang o vi
tri Max-1 thi tang ve vi tri 0
            Q.count++; //tang so phan tu len
      }
}

int Pop_Circular(Queue &Q) //Loai bo phan tu khoi dau hang doi vong
{
      if (Isempty(Q)) cout << "Hang doi rong !";
      item x = Q.Data[Q.Front];
      Q.Front = (Q.Front++) % Max; // tang vi tri phan dau tien len,
neu dang o Max-1 thi ve 0
      Q.count--;//giam so phan tu xuong
      return x; //tra ve phan tu lay ra
}


void Input(Queue &Q) //nhap hang doi
{
      int n;
      item x;
      do
      {
            cout << "Nhap so phan tu cua Queue  :"<< Max;
            cin>>n;
      } while (n>Max || n<1);
      for (int i = 0; i<n; i++)
      {
            cout<<"Nhap phan tu thu : "<< i + 1;
            cin>>x;
            Push(Q, x);
            Push_Circular(Q, x); //hang vong
      }
}

void Output(Queue Q)
{
      if (Isempty(Q)) cout << "Hang doi rong !";
      else
```

```
	{
		for (int i = Q.Front; i <= Q.Rear; i++)
			//for (int i=Q.Front, j=0; j<Q.count; j++, i =
(i++) % Max) //hang vong
			cout << Q.Data[i];
		cout << endl;
	}
}

int main()
{
	Queue Q;
	Init(Q);
	Input(Q);
	Output(Q);

	int lua_chon;
	cout << "Moi ban chon phep toan voi DS LKD:";
	cout << "\n1: Kiem tra Queue rong";
	cout << "\n2: Kiem tra Queue day";
	cout << "\n3: Them phan tu vao Queue";
	cout << "\n4: Xoa phan tu trong Queue";
	cout << "\n5: Xuat Queue";
	cout << "\n6: Thoat";
	do
	{
		cout << "\nBan chon: ";
		cin>>lua_chon;
		switch (lua_chon)
		{
		case 1:
		{
			if (Isempty(Q)) cout << "Queue rong !";
			else cout << "Queue khong rong !";
			break;
		}
		case 2:
		{
			if (Isfull(Q)) cout << "Queue day !";
			else cout << "Queue chua day !";
			break;
		}
		case 3:
		{
			item x;
			cout << "Nhap phan tu can chen vao Queue: ";
			cin>>x;
```

```
                Push(Q, x);
                //Push_Circular(Q,x); hang vong
                break;
            }
            case 4:
            {
                Pop(Q);
                //Pop_Circular(Q); hang vong
                break;
            }
            case 5:
            {
                Output(Q);
                break;
            }
            case 6: break;
            }
      } while (lua_chon != 6);
      return 0;
}
```

Bài tập về Queue

1. Nhập N ≤ 1000 số nguyên từ bàn phím. Với mỗi số nhập vào, thêm vào Stack S nếu là số chẵn, ngược lại thêm vào Queue Q nếu là số lẻ. Xuất các phần tử có trong S và Q ra màn hình, dòng đầu tiên là các phần tử của S, dòng tiếp theo là các phần tử của Q.