**BÁO CÁO ĐỒ ÁN CUỐI KÌ**

**Môn học:** cấu trúc dữ liệu – giải thuật

**Lớp: 09.**

**Nhóm: 23.**

1. Nguyễn Đức Anh – 21110864.

**Đề tài:** đề tài bổ sung 1: Viết chương trình mô phỏng dạng hình ảnh cho các cấu trúc dữ liệu stack, queue, dequeue, link list, circle list với các phép toán insert, delete.

**1.Stack**

//khởi tạo

Void init()

{

Node\* top;

Top = nullptr;

}

//kiểm tra rỗng

Bool IsEmpty()

{

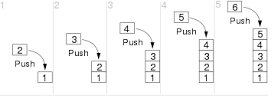
If(top == nullptr)

Return true;

Return false;

}

**1.1 insert**



Void insert(node\* p)

{

If(IsEmpty())

{

Top = p;

Return;

}

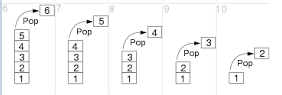
Node\* temp = top;

Top = p;

Top -> next = temp;

}

**1.2 delete**



Void delete()

{

If(IsEmpty())

{

Return;

}

Node\* temp = top -> next;

Delete(top);

Top = temp;

}

**2.Queue**

//khởi tạo

Void init()

{

Node\* front;

Node\* rear;

Front = rear = nullptr;

}

//kiểm tra rỗng

Bool IsEmpty()

{

If(rear == nullptr)

Return true;

Return false;

}

**2.1 insert**

Void insert(node\* p)

{

If(IsEmpty())

{

Front = rear = p;

Return;

}

Rear -> next = p;

rear = p;

}

**2.2 delete**

Void delete()

{

If(IsEmpty())

{

Return;

}

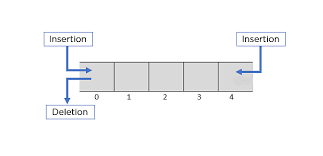
Node\* temp = front;

Front = front -> next;

Delete(temp);

}

**3.Dequeue**

****

//khởi tạo

Void init()

{

Node\* leftmost;

Node\* rightmost;

Leftmost = rightmost = nullptr;

}

//kiểm tra rỗng

Bool IsEmpty()

{

If(rightmost == nullptr)

Return true;

Return false;

}

**3.1 insert**

Void insert(node\* p, bool in)

{

If(IsEmpty())

{

Leftmost = rightmost = p;

Return;

}

If(in == true) // đưa vào left

{

Node\* temp = Leftmost;

Leftmost = p;

Leftmost -> next = temp;

}

Else // đưa vào right

{

Rightmost -> next = p;

Rightmost = Rightmost -> next;

}

}

**3.2 Delete**

Void delete()

{

If(IsEmpty())

{

Return;

}

Node\* temp = Leftmost;

Leftmost = Leftmost -> next;

Delete(temp);

}

**4. Link list**

****

//khởi tạo

Void init()

{

Node\* Head;

head = nullptr;

}

//kiểm tra rỗng

Bool IsEmpty()

{

If(Head == nullptr)

Return true;

Return false;

}

**4.1 Insert**

Void insert(node\* p, int k)

{

If( IsEmpty() )

{

Head = p;

Head ->next = nullptr;

Return;

}

If(k == 0)

{

Node\* temp = Head;

Head = p;

Head -> next = temp;

}

Else

{

Node\* temp = Head;

For(int i = 1; i < k; i++)

Temp = temp -> next;

If(temp -> next == nullptr)

{

Temp -> next = p;

P -> next = nullptr;

Return;

}

Node\* tempn = temp -> next;

Temp -> next = p;

p -> next = tempn;

}

}

**4.2 Delete**

Void delete(node\* p, int k)

{

If( IsEmpty() )

{

Return;

}

Node\* temp = Head;

If( k == 0)

{

Head = Head -> next;

Delete(temp);

Return;

}

For(int i = 1; i < k-1; i++)

temp = temp -> next;

node\* tempn = temp -> next -> next;

delete(temp -> next);

temp -> next = tempn;

}

**5. Circle list**

****

//khởi tạo

Void init()

{

Node\* Head;

Head = nullptr;

}

//kiểm tra rỗng

Bool IsEmpty()

{

If(Head == nullptr)

Return true;

Return false;

}

**5.1. Insert**

Void insert(node\* p, int k)

{

If( IsEmpty() )

{

Head = p;

Head ->next = Head;

Return;

}

Node\* temp = Head;

For(int i = 1; i < k; i++)

Temp = temp -> next;

Node\* tempn = temp -> next;

Temp -> next = p;

p -> next = tempn;

}

**5.2 Delete**

Void delete(node\* p, int k)

{

If( IsEmpty() )

{

Return;

}

Node\* temp = Head;

For(int i = 1; i < k-1; i++)

temp = temp -> next;

if(temp == Head && temp -> next == Head)

{

Head = nullptr;

Delete(temp);

Return;

}

node\* tempn = temp -> next -> next;

delete(temp -> next);

temp -> next = tempn;

}