**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN : CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**NHÓM 6 LỚP 09**

|  |  |
| --- | --- |
| **Họ Tên** | **Mã số sinh viên** |
| **Nguyễn Dũng Hiệp** | **21110883** |
| **Lê Thanh Hùng** | **21110479** |
| **Phan Nguyên Định** | **21110423** |

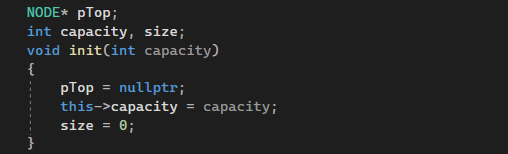
**Chi tiết nhiệm vụ**

|  |  |
| --- | --- |
| **Nguyễn Dũng Hiệp** | **- Lên ý tưởng làm chương trình mô phỏng**  **- Thực hiện làm các file: queue.h, queue.cpp, deque.h, deque.cpp, global.h, global.cpp, linklist.h, linklist.cpp, circlelist.h, circlelist.cpp, Simulation.cpp, DataStructure.h**  **- Viết thêm các hàm cho thư viện graphics, file graphics.h, graphics.cpp**  **- Thêm comment giải thích cho các file code**  **- Tổng hợp bài của các thành viên và điều chỉnh thành chương trình hoàn chỉnh** |
| **Phan Nguyên Định** | **- Thực hiện làm các file topo.h, topo.cpp**  **- Làm báo cáo thuật toán topo** |
| **Lê Thanh Hùng** | **Thực hiện làm các file stack.cpp, stack.h**  **- Làm báo cáo giải thích thuật toán thêm, xóa cho các cấu trúc dữ liệu** |

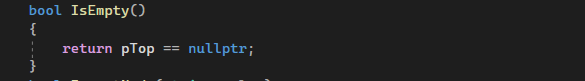
**I. STACK**

**1. CẤU TRÚC DỮ LIỆU STACK (**struct LINKEDSTACK)

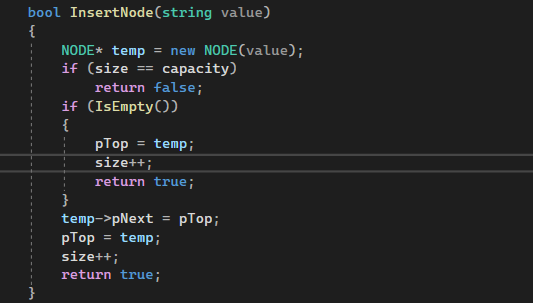
Hàm khởi tạo

****

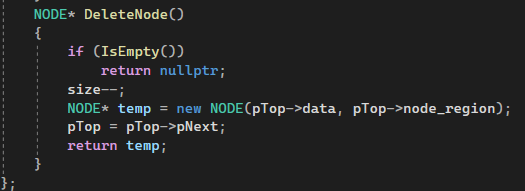
Hàm kiểm tra stack rỗng

****

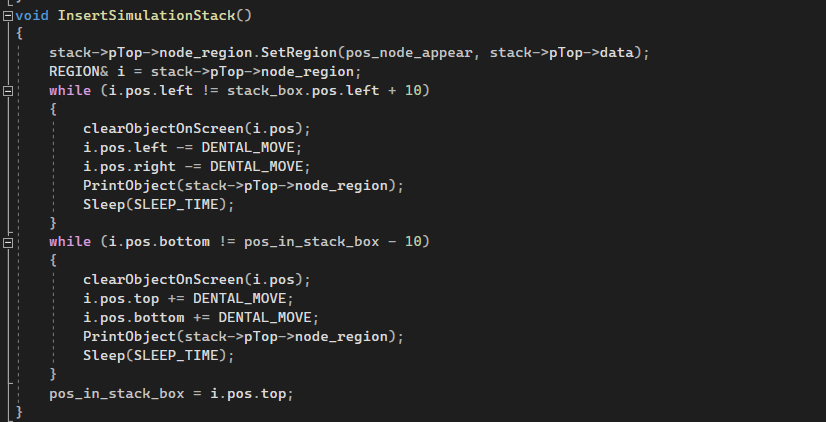
Hàm thêm NODE vào stack

****

Hàm xóa NODE khỏi stack

****

**2 . CHƯƠNG TRÌNH MÔ PHỎNG STACK**

**a )** **MÔ PHỎNG THÊM NODE (**InsertSimulationStack())

Ta đặt vị trí cho NODE mới bằng cú pháp:

stack->pTop->node\_region.SetRegion(pos\_node\_appear, stack->pTop->data);

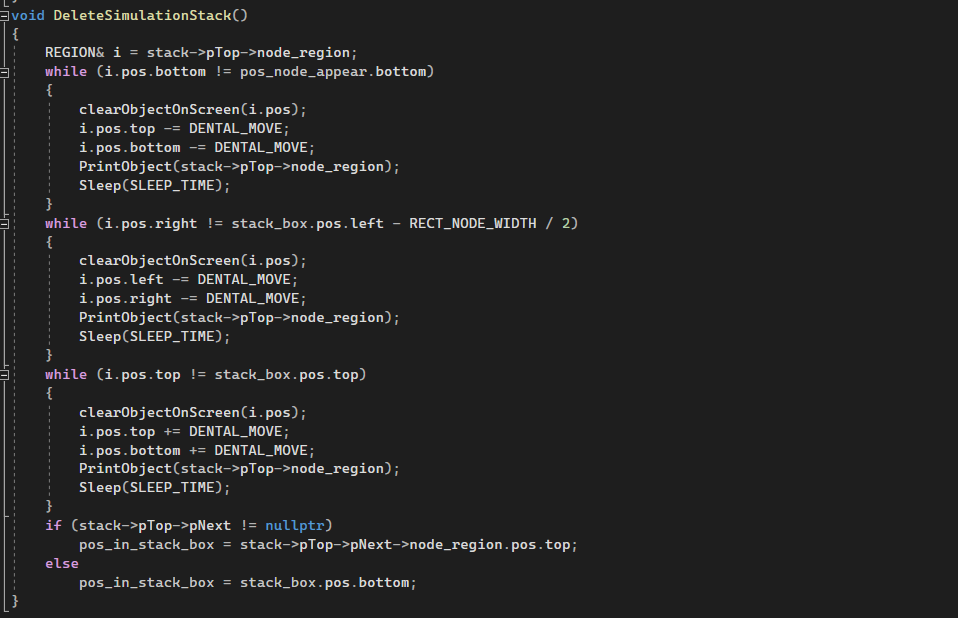
và tham chiếu đến NODE mới thêm trên consolo

REGION& i = stack->pTop->node\_region;

NODE sẽ được di chuyển vào stack thông qua 2 vòng lặp while như trên hình và sau đó kết thúc quá trình thêm node đầu tiên bằng cách cập nhật vị trí thêm NODE tiếp theo

pos\_insert\_node\_stack = i.pos.top;

**e ) MÔ PHỎNG XÓA NODE (**DeleteSimulationStack())



Chương trình sẽ chạy các vòng lập while như trên để tiến mô phỏng quá trình đưa NODE ra khỏi stack và sau đó cập nhật vị trí NODE tiếp theo

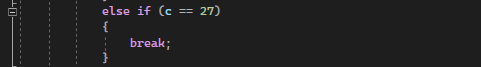
if (stack->pTop->pNext != nullptr)

pos\_insert\_node\_stack = stack->pTop->pNext->node\_region.pos.top;

else

pos\_insert\_node\_stack = stack\_box.pos.bottom;

**f ) THOÁT CHƯƠNG TRÌNH**



Chương trình mô phỏng stack sẽ kết thúc nếu ta ấn ESC từ bàn phím.

**II. QUEUE**

**1. CẤU TRÚC DỮ LIỆU QUEUE (struct QUEUE)**

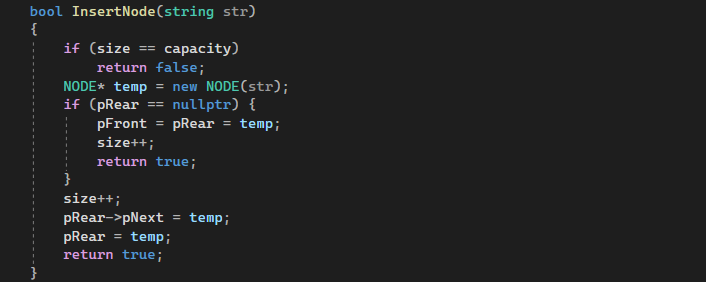
Hàm khởi tạo queue

****

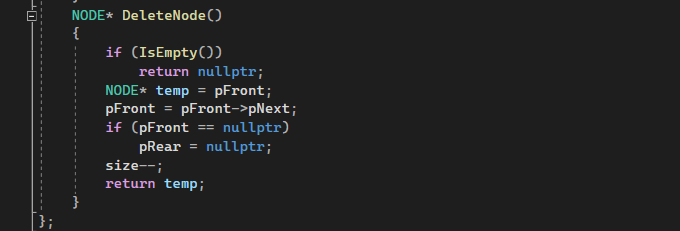
Hàm kiểm tra queue rỗng

****

Hàm thêm NODE vào queue

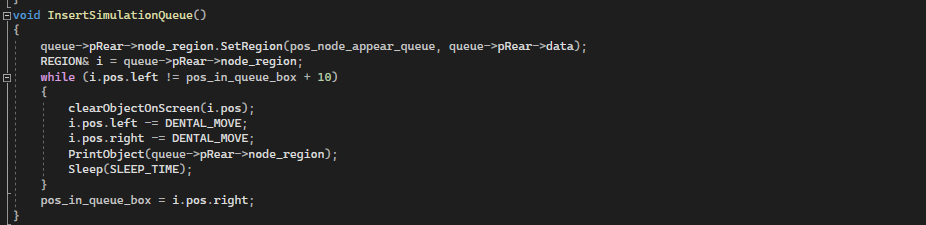
****

Hàm xóa NODE khỏi queue

****

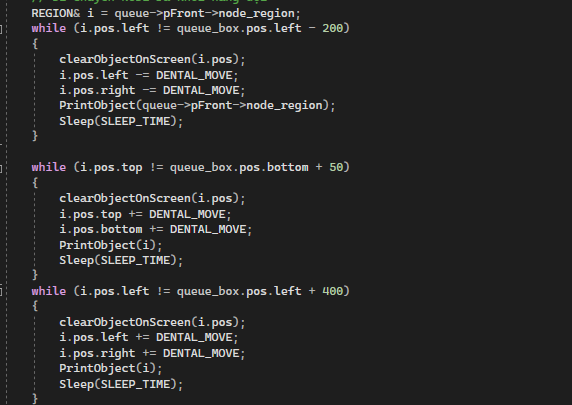
**2. CHƯƠNG TRÌNH MÔ PHỎNG QUEUE**

**a ) HÀM MÔ PHỎNG THÊM NODDE VÀO QUEUE**

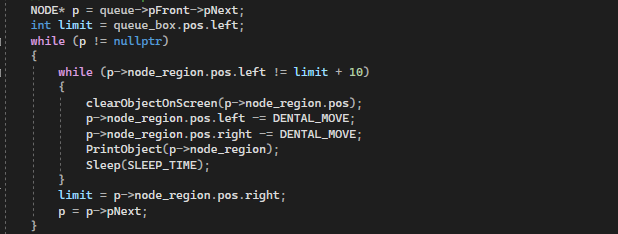
****

**b ) HÀM MÔ PHỎNG XÓA NODE KHỎI QUEUE**

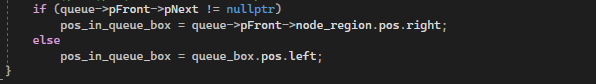
Di chuyển NODE ra khỏi hàng đợi

****

Đưa các NODE phía sau lên trước

****

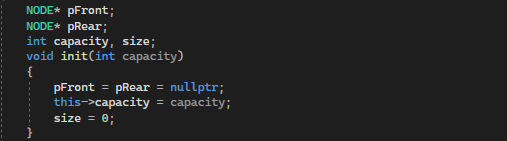
Cập nhập vị trí thêm NODE tiếp theo

****

**III. DEQUEUE**

**1. CẤU TRÚC DỮ LIỆU DEQUEUE (struct DEQUEUE)**

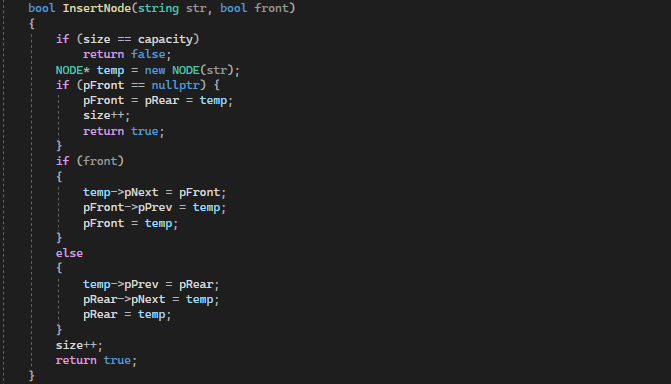
Hàm khởi tạo dequeue

****

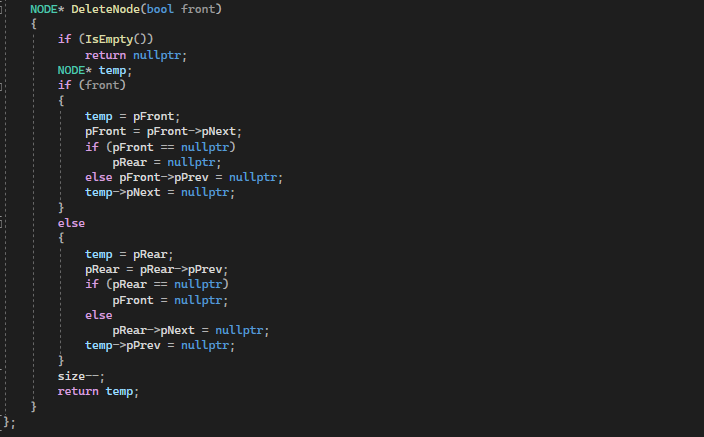
Kiểm tra dequeue rỗng

**C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (257).png**

Hàm thêm NODE vào dequeue

****

Hàm xóa NODE khỏi dequeue

****

**2. CHƯƠNG TRÌNH MÔ PHỎNG DEQUEUE**

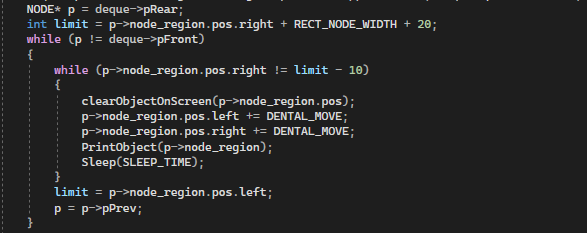
**a ) MÔ PHỎNG THÊM NODE VÀO DEQUEUE**

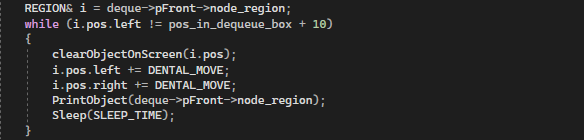
**\* Thêm NODE vào phía trước dequeue**

Xác định vị trí thêm NODE trên console

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (265).png

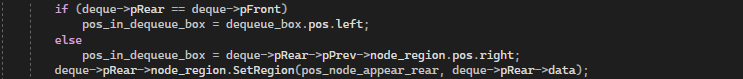
Di chuyển các NODE ra phía sau để trống vị trí thêm NODE

****

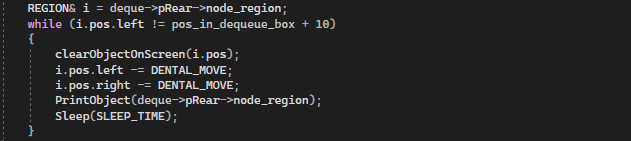
Di chuyển NODE vào hàng đợi****

**\* Thêm NODE vào phía sau dequeue**

Xác định vị trí thêm NODE

****

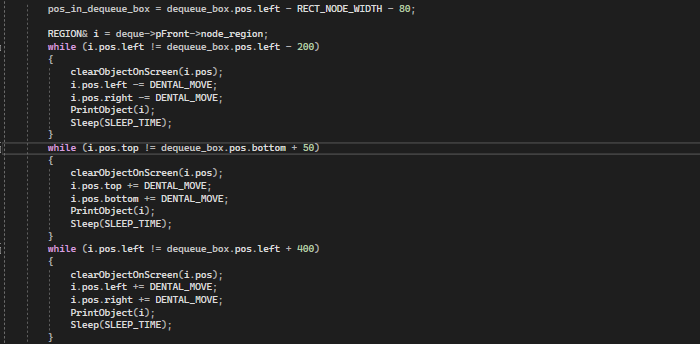
Di chuyển NODE vào hàng đợi

****

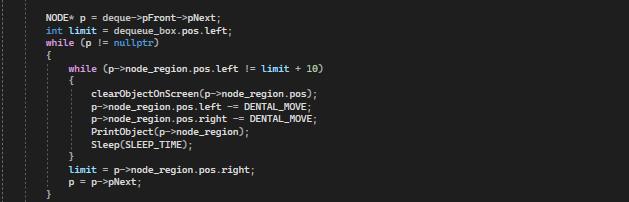
**b ) MÔ PHỎNG XÁO NODE KHỎI DEQUEUE**

**\* Xóa NODE phía trước dequeue**

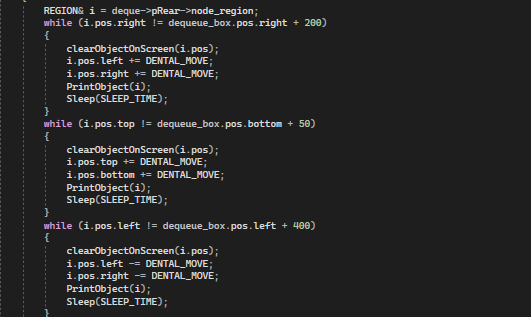
Đưa NODE bị xóa ra khỏi hàng đợi

****

Di chuyển các NODE phía sau lên trước

****

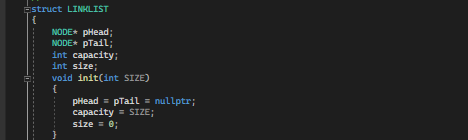
**\* Xóa NODE phía sau dequeue**

****

**IV. LINKLIST**

**1. CẤU TRÚC DỮ LIỆU LINKLIST**

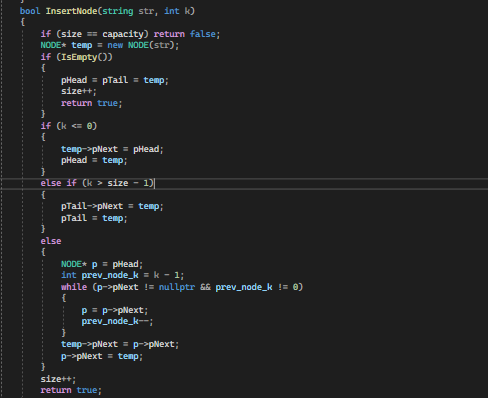
Hàm khởi tạo LINKLIST

****

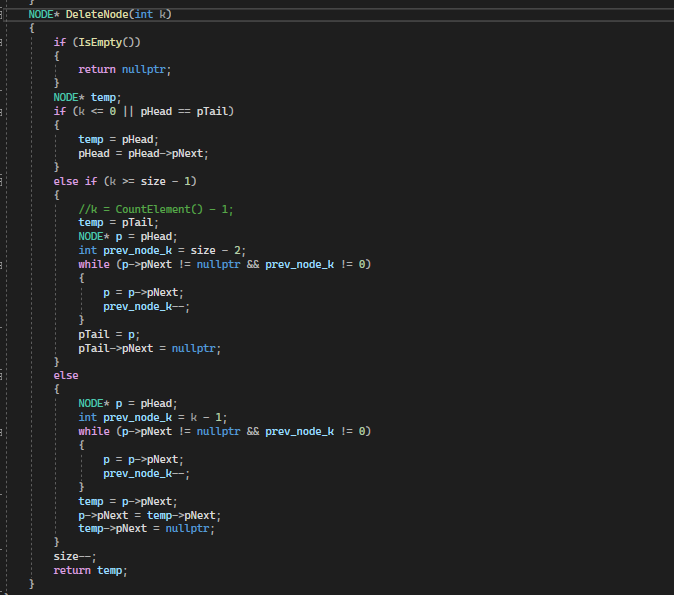
Hàm kiểm tra LINKLIST rỗng

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (278).png

Hàm thêm NODE vào LINKLIST



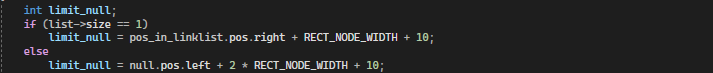
Hàm xóa NODE khỏi LINKLIST



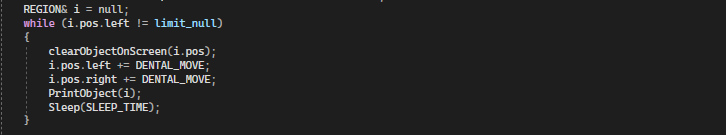
**2. CHƯƠNG TRÌNH MÔ PHỎNG LINKLIST**

a ) MÔ PHỎNG THÊM NODE (void InsertSimulationList(int k))

Đặt vị trí cho con trỏ NULL



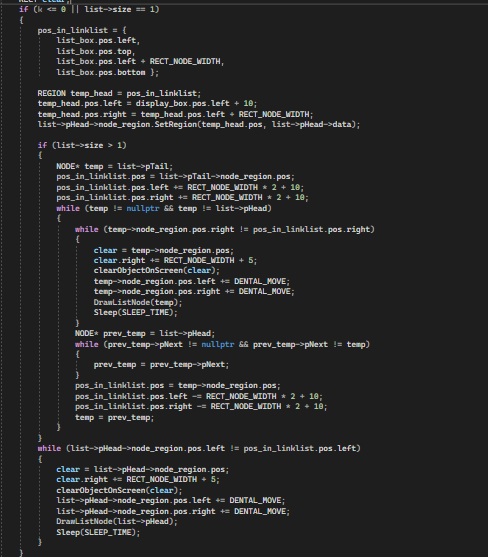
Di chuyển con trỏ NULL



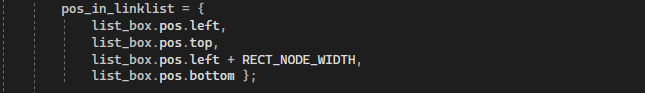
Lưu vị trí xóa đối tượng

RECT clear;

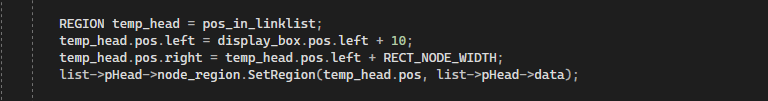
**\*Thêm NODE vào đầu list**



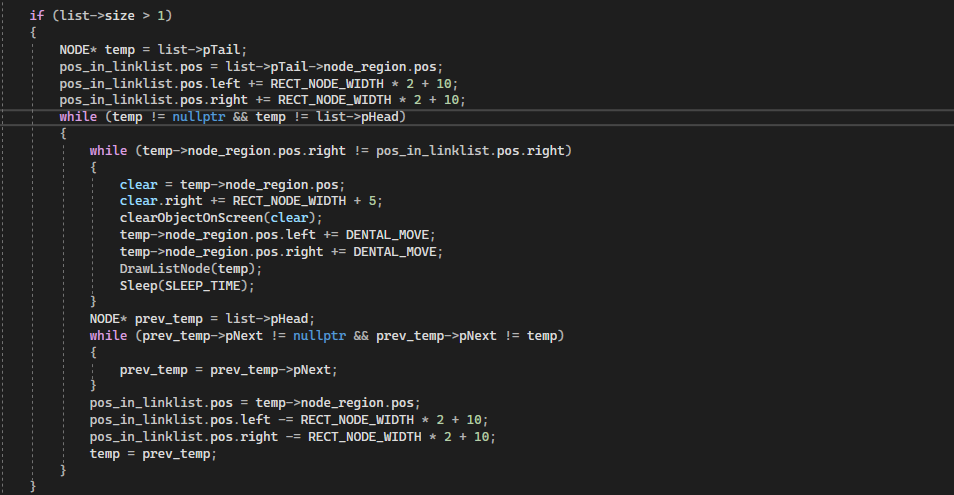
Trong đó, để xác định vị trí thêm NODE trên màn hình console ta dùng



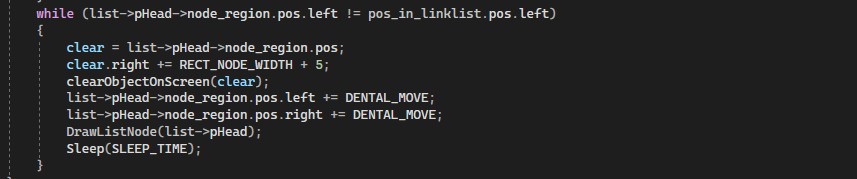
Sau đó tạo ví trí ban đầu của NODE và vị trí NODE di chuyển tới



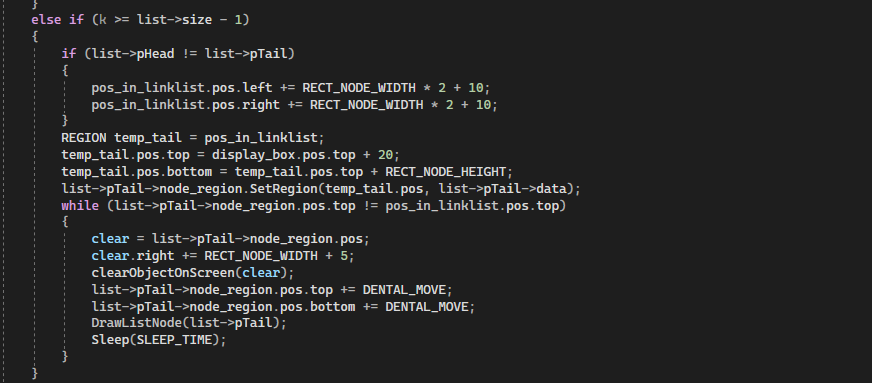
Nếu list ban đầu không trống di chuyển các NODE ra phía sau làm trống vị trí đầu tiên để thêm NODE mới



Sau đó Di chuyển NODE mới vào vị trí trong list



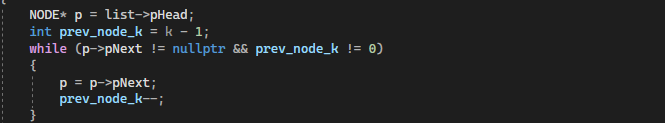
\*THÊM NODE VÀO CUỐI LIST



Tương tự như thêm node vào đầu list thì ta cũng sẽ Đặt lại vị trí thêm NODE nếu list ban đầu không trống sau đó Tạo vị trí ban đầu của NODE và vị trí NODE di chuyển tới và cuối cùng là Di chuyển NODE mới vào vị trí trong list

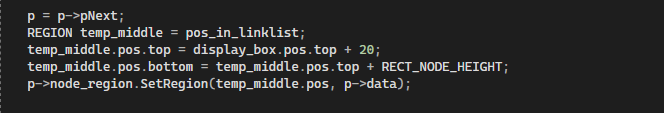
\*THÊM NODE VÀO GIỮA LIST

Tìm NODE phía trước NODE được thêm

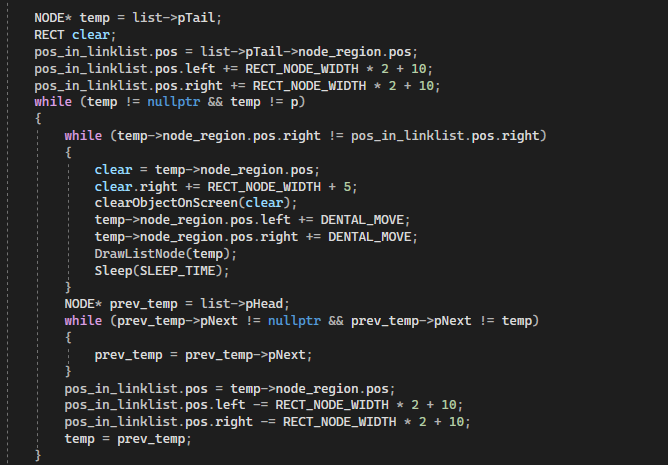


Trỏ đến NODE được thêm: p = p->pNext;

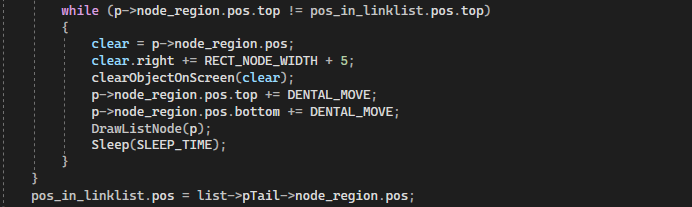
Sau đó Tạo ví trí ban đầu của NODE và vị trí NODE di chuyển tới



Di chuyển các NODE để tạo chỗ trống thêm NODE mới

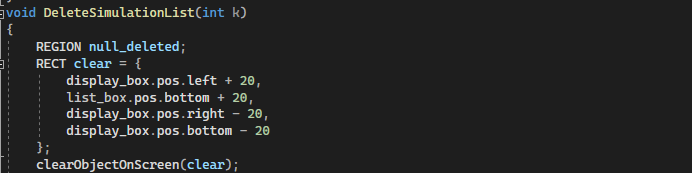


Sau đó di chuyển NODE mới vào vị trí trong list



b ) MÔ PHỎNG XÓA NODE (void DeleteSimulationList(int k))

Xóa khu vực hiện thị NODE được xóa trước đó



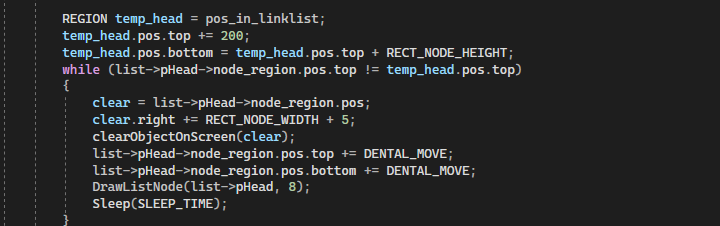
\*XÓA NODE HEAD

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (312).png

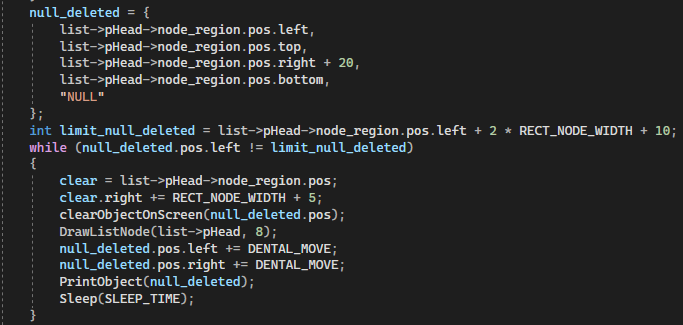
Đầu tiên là xác định vị trí trên console của NODE bị xóa

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (313).png

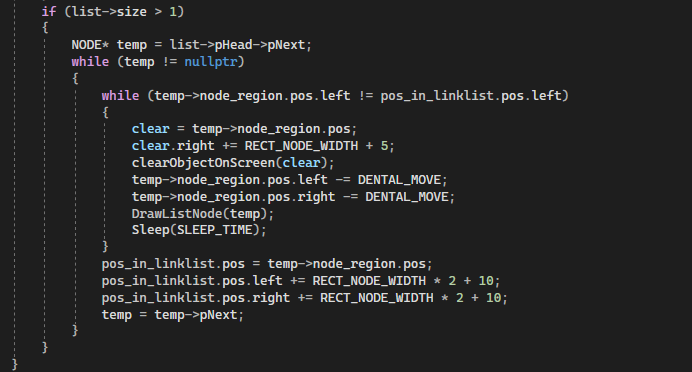
Sau đó di chuyển NODE Head ra khỏi list



Tiếp đó Vẽ con trỏ NULL cho NODE bị xóa



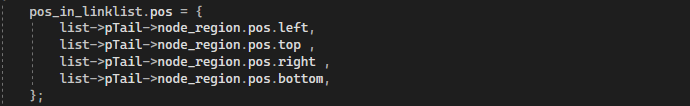
Nếu số phần tử của list > 1 thì Di chuyển các NODE ra phía trước lấp vị trí vừa xóa NODE



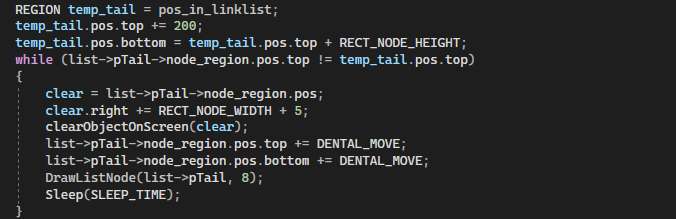
\*XÓA NODE CUỐI

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (321).png

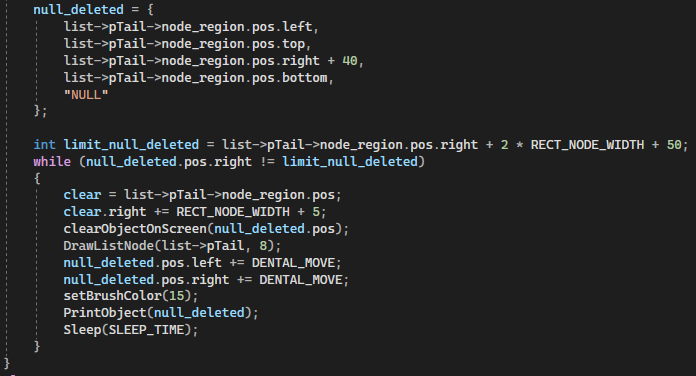
Xác định vị trí trên console của NODE bị xóa



Di chuyển NODE Head ra khỏi list

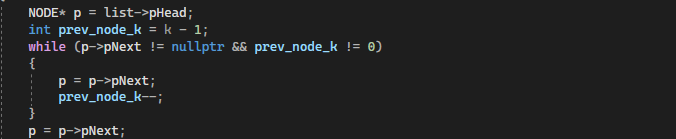


Sau đó vẽ con trỏ NULL cho NODE bị xóa

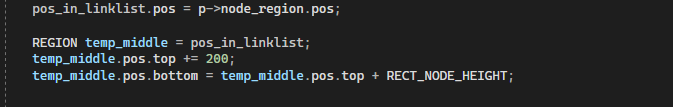


\*XÓA NODE GIỮA

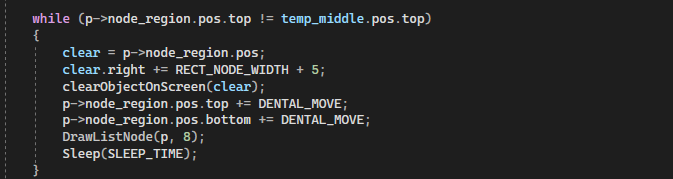
Tìm NODE sẽ bị xóa



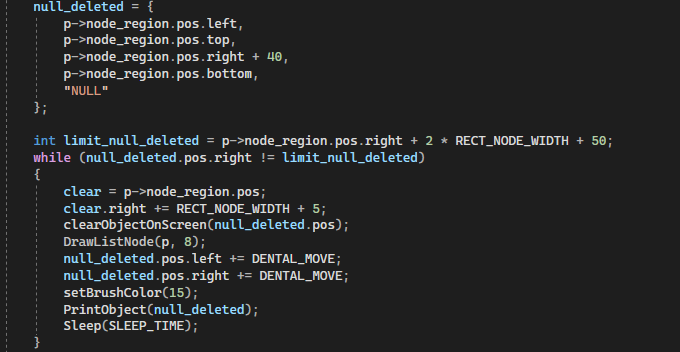
Xác định vị trí trên console của NODE bị xóa



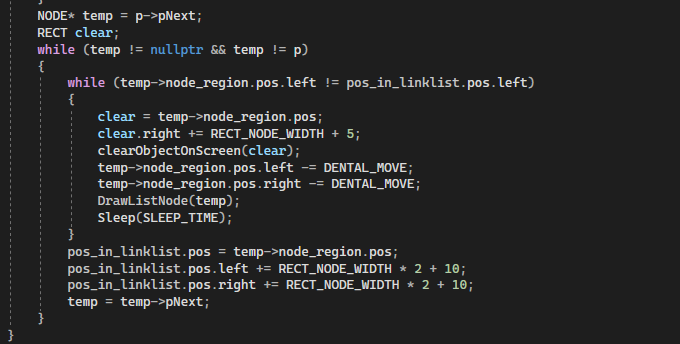
Di chuyển NODE bị xóa ra khỏi list



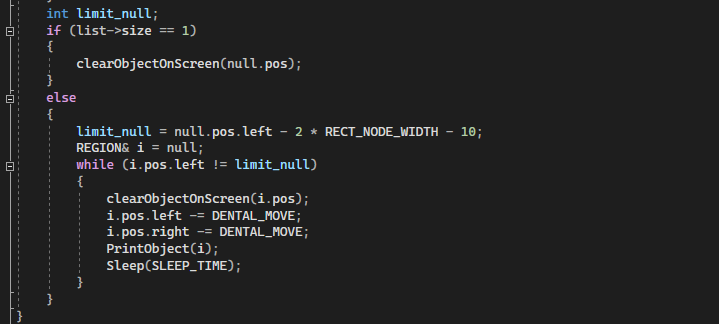
Vẽ con trỏ NULL cho NODE bị xóa



Di chuyển các NODE phía sau NODE bị xóa lên trước



Xóa hoặc di chuyển con trỏ NULL dựa vào list có trống hay không sau khi xóa



**V.CIRCLELIST**

**1.CẤU TRÚC DỮ LIỆU CIRCLELIST**

**2.CHƯƠNG TRÌNH MÔ PHỎNG CIRCLELIST**

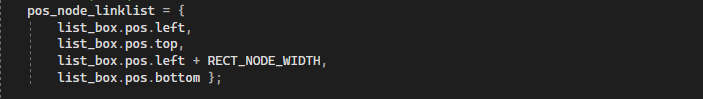
**a ) MÔ PHỎNG THÊM NODE VÀO CIRCLELIST**

(void InsertSimulationCircleList(int k))

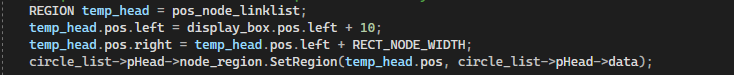
\* THÊM NODE VÀO ĐẦU CIRCLELIST

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (340).png

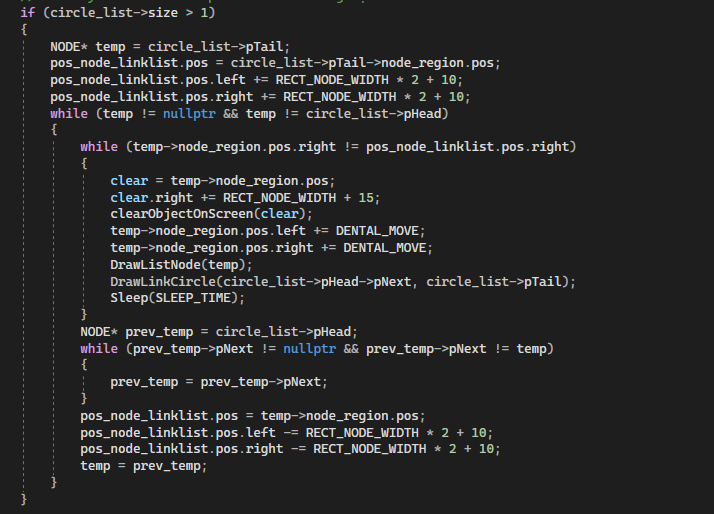
Xác định vị trí thêm NODE trên màn hình console



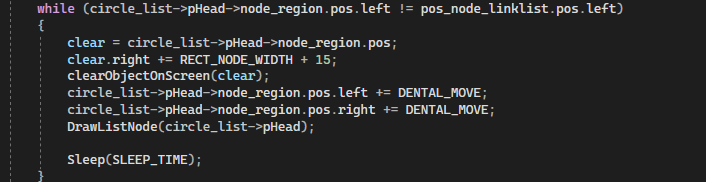
Tạo ví trí ban đầu của NODE và vị trí NODE di chuyển tới



Nếu list ban đầu không trống di chuyển các NODE ra phía sau làm trống vị trí đầu tiên để thêm NODE mới



Di chuyển NODE mới vào vị trí trong list



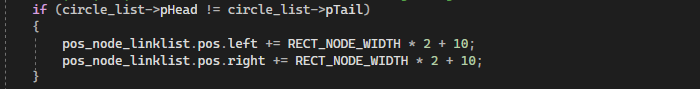
Vẽ liên kiết giữa NODE đầu và NODE cuối

DrawLinkCircle(circle\_list->pHead, circle\_list->pTail);

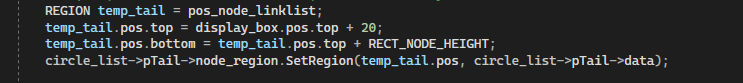
\* THÊM NODE VÀO CUỐI CIRCLELIST

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (349).png

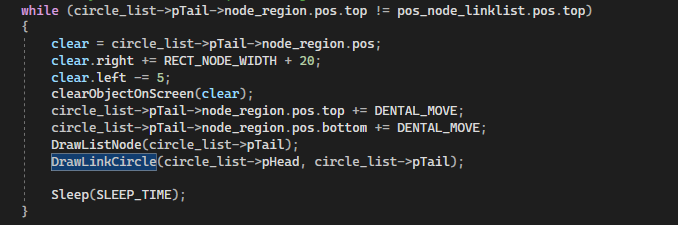
Đặt lại vị trí thêm NODE nếu list ban đầu không trống



Tạo vị trí ban đầu của NODE và vị trí NODE di chuyển tới

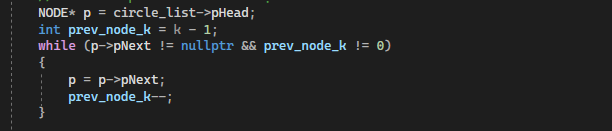


Di chuyển NODE mới vào vị trí trong list



\* THÊM NODE VÀO GIỮA CIRCLELIST

Tìm NODE phía trước NODE được thêm



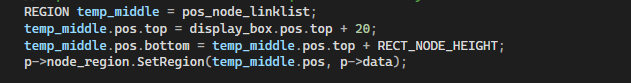
Đặt lại vị trí thêm NODE

C:\Users\Thanh Hung\Pictures\Screenshots\Screenshot (356).png

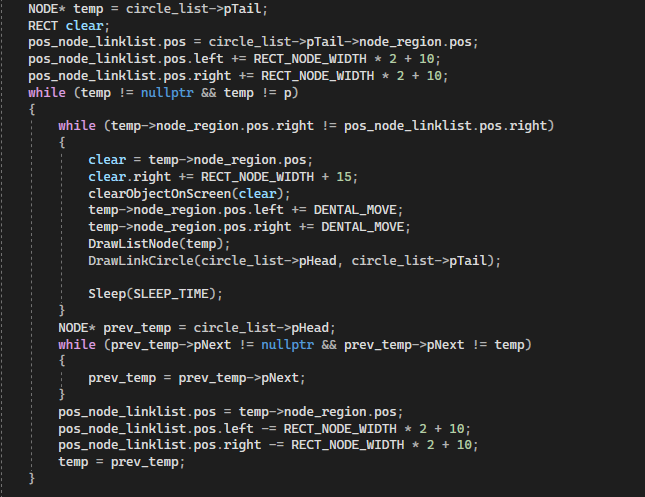
Trỏ đến NODE được thêm

p = p->pNext;

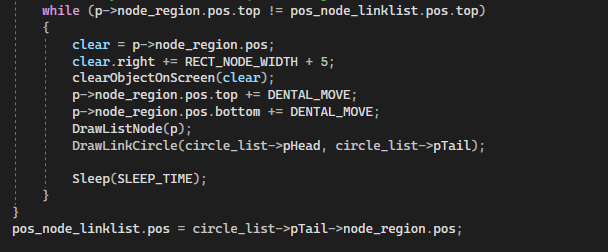
Tạo ví trí ban đầu của NODE và vị trí NODE di chuyển tới



Di chuyển các NODE để tạo chỗ trống thêm NODE mới



Di chuyển NODE mới vào vị trí trong list



b ) MÔ PHỎNG XÓA NODE (void DeleteSimulationCircleList(int k))

Xóa khu vực hiện thị NODE được xóa trước đó



\* XÓA NODE ĐẦU

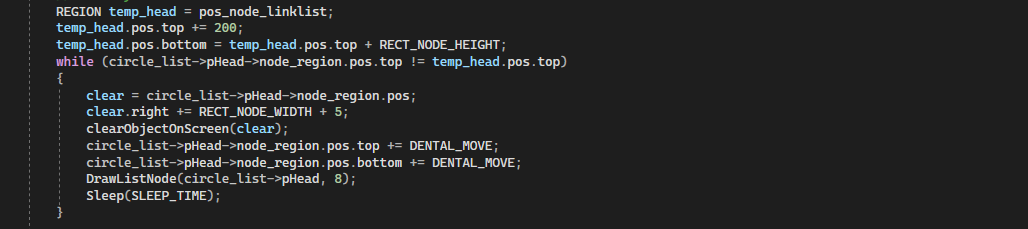
if (k <= 0 || circle\_list->pHead == circle\_list->pTail)

{

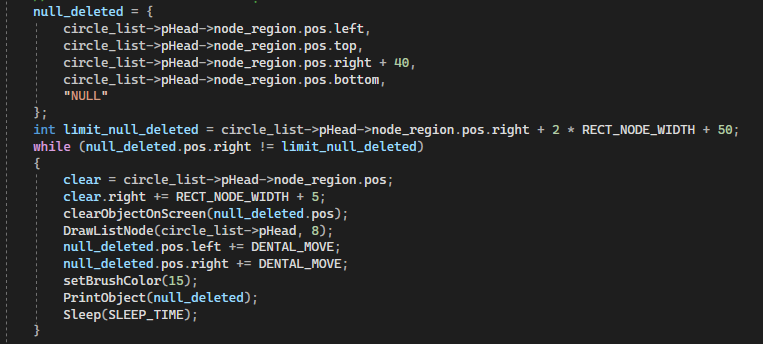
Xác định vị trí trên console của NODE bị xóa

pos\_node\_linklist.pos = circle\_list->pHead->node\_region.pos;

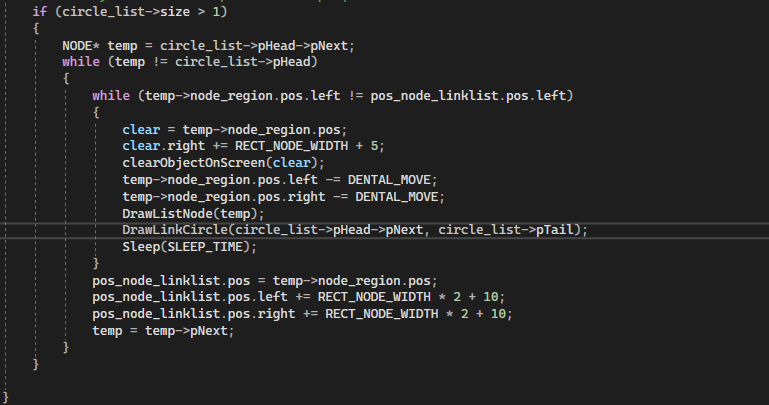
Di chuyển NODE Head ra khỏi list



Vẽ con trỏ NULL cho NODE bị xóa

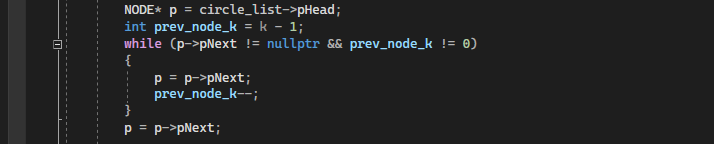


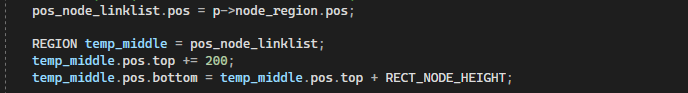
Nếu số phần tử của list > 1 di chuyển các NODE ra phía trước lấp vị trí vừa xóa NODE



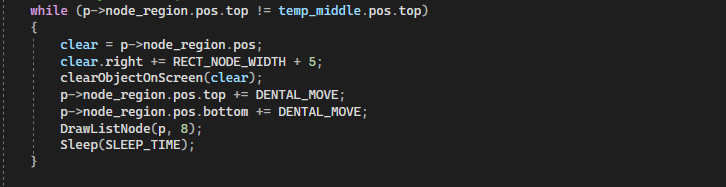
\* XÓA NODE GIỮA

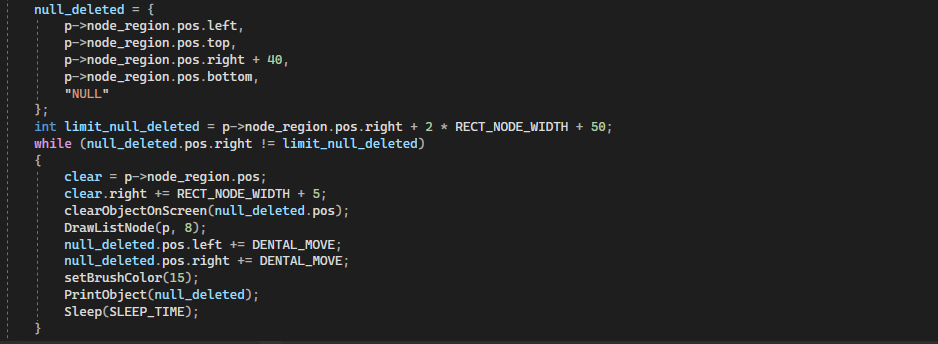
Tìm NODE sẽ xóa

Xác định vị trí trên console của NODE bị xóa

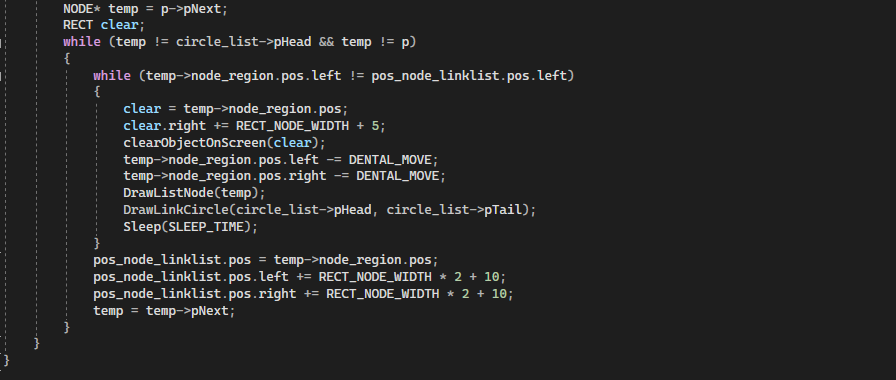


Di chuyển NODE bị xóa ra khỏi list



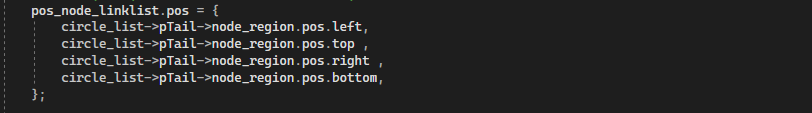
Vẽ con trỏ NULL cho NODE bị xóa

Di chuyển các NODE phía sau NODE bị xóa lên trước

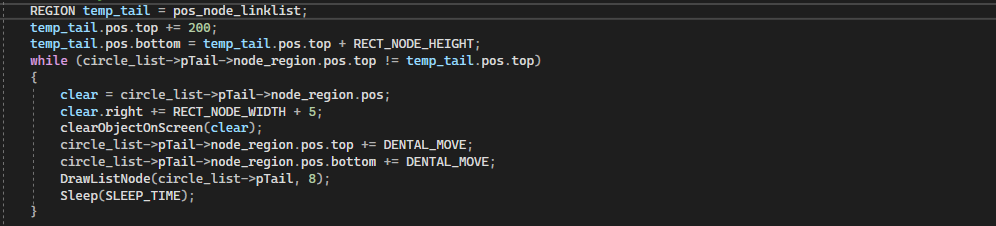


\* XÓA NODE CUỐI

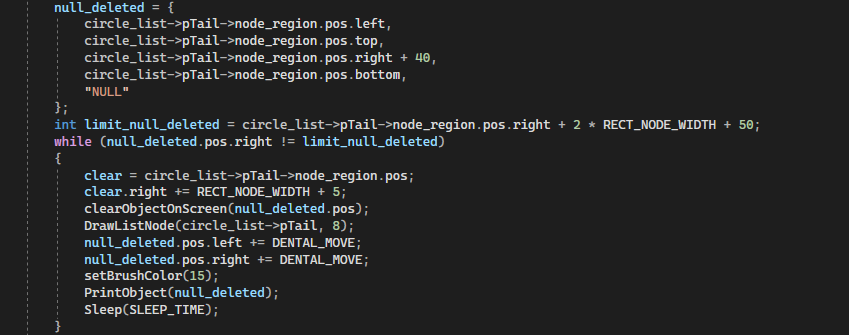
Xác định vị trí trên console của NODE bị xóa



Di chuyển NODE Head ra khỏi list



Vẽ con trỏ NULL cho NODE bị xóa



**BÀI TOÁN SẮP XẾP TÔ-PÔ:**

Thứ tự Tô-pô của một đồ thị có hướng là một thứ tự sắp xếp của các đỉnh sao cho với mọi cung từ đỉnh u đến đỉnh v trong đồ thị, u luôn nằm trước v.

Điều kiện tồn tại thứ tự Tô-pô:

Một đồ thị tồn tại thứ tự Tô-pô thì đồ thị đó phải là đồ thị có hướng và không có chu trình *(Directed Acyclic Graph - DAG)*.

Thứ tự Tô-pô không nhất thiết phải là duy nhất. Có thể có một số thứ tự Tô-pô khác nhau trong một đồ thị.

Bài toán xác định thứ tự Tô-pô gọi là Bài toán sắp xếp Tô-pô.

**Thuật toán:**

Cấu trúc dữ liệu:

* Struct SUCCESOR :
* Struct NODE: COUNT: số cạnh đi vào node
* Mảng QLINK : mảng hàng đợi.
* Mảng A[] mảng chứa các node.
* Mảng P[] (SUCCESSOR)
* Mảng out[] mảng lưu thứ tự topo.

**Input và đặt các giá trị của NODE từ input :**

cin >> N >> num\_relation;

int number = N;

for (int i = 1; i <= N; i++)

{

A[i].COUNT = 0;

A[i].TOP = nullptr;

}

for (int i = 0; i < num\_relation; i++)

{

int x, y;

cin >> x >> y;

A[y].COUNT = A[y].COUNT + 1;

P = (SUCCESSOR\*)malloc(sizeof(SUCCESSOR));

P->SUC = y;

P->next = A[x].TOP;

A[x].TOP = P;

}

**Cập nhật NODE successor vào hàng đợi :**

QLINK[0] = 0;

for (int i = 1; i <= number; i++)

{

if (A[i].COUNT == 0)

{

QLINK[R] = i;

R = i;

}

}

**Kiếm tra và đẩy các NODE mà COUNT = 0 vào hàng đợi**

**Duyệt các successor của NODE lấy ra từ hàng đợi**

**Nếu COUNT[k] -- == 0**

**nghĩa là NODE successor đó chỉ có 1 cạnh đi vào nó**

**thì đẩy NODE successor vào hàng đợi**

**Nếu không duyệt đến successor tiếp theo**

for (F = QLINK[0]; F != 0; F = QLINK[F])

{

out[nout++] = F;

N--;

for (P = A[F].TOP; P != nullptr; P = P->next)

{

A[P->SUC].COUNT--;

if (A[P->SUC].COUNT == 0)

{

QLINK[R] = P->SUC;

R = P->SUC;

}

}

}

**Xử lý topo và xuất ra màn hình :**

out[nout++] = 0;

if(N==0)

{

for (int i = 0; i < number; i++)

cout << out[i] << " ";

}