VELLORE INSTITUTE OF TECHNOLOGY

(Estd. u/s 3 of UGC Act 1956)

Vellore - 632 014, Tamil Nadu, India

School of Computer Science and Engineering

Winter semester 2019-20

CSE2003 Data Structures and Algorithms

J Component

Report

on

Snake Game & Tic Tac Toe

By

Jeet Bangoria – 19BCE0874

Submitted to:

Prof. Seetha R

ACKNOWLEDGEMENT

Our special thanks go to one and only **Prof. Seetha R.** who supported us throughout our journey. This project consumed huge amount of work, research and dedication.

I would also like to acknowledge VIT for giving the candidates an opportunity to carry out our studies at the institute.

Table of Contents

Abstract

The project aims to implement games using Linked Lists. A linear linked list is like a clothes line on which the data structures hang sequentially. A head pointer addresses the first element of the list, and each element points at a successive element, with the last element having a link value NULL.

Based on the number of link parts, the linked lists may be singly linked lists or doubly linked lists. The linked list in which there is a data part and an address part is called a singly linked list and list having a data pert and two address parts is called a doubly linked list.

The main operations I can do on a linked list are insertion, deletion and display the contents of the list.

Introduction

Linked List

A linked list is a data structure which is dynamic in nature. A linked list is a linear data structure which involved individual elements. These individual elements are called nodes. Each node hold information as Ill as the address of the next node. As I discuss the properties of a linked list, I will come across the various advantages of a linked list over arrays. Linked List involves the implementation of structures. Structures involve user defined data types which help the user design the desired output.

Properties of a Linked List

- When a user is unaware of the space or memory required, linked list can solve this issue. When I had an unknown number of items to deal with, a linked list can be maximized or minimized as I work along with it. Functions like adding or deleting a node in the beginning, ending or in the middle of the linked list can constantly alter the linked list. A fixed amount of memory need not be allocated.
- The allocation and deallocation of memory is less expensive when compared to arrays.
- During insertion or deletion of elements, the elements of a linked list need not be shifted.
- Reduces a lot of memory wastage. In arrays, I specified a certain amount of memory, but only half of it has been assigned with values, the other half of the memory goes to waste. But in the case of a linked list, only the required amount of memory is allocated.

Singly Linked List

A singly linked list is a type of linked list where the node only stores content and a reference to the next node but it does not hold any information related to the previous node of that particular linked list. The last node of a singly linked list does not give reference to any other node. The first node is considered to be the reference node and also called as the head node for prior understanding.

Methodology

Header files used

1. graphics.h
2. dos.h
3. alloc.h
4. conio.h
5. stdio.h
6. stdlib.h

<graphics.h>

- cleardevice():

  It clears the screen in graphics mode and sets the current position to (0,0).

- initgraph();

  It is used to initialize the graphics mode.

- closegraph():

  It closes the graphics mode, deallocates memory allocated my graphics system, restores the screen to how it was after 'initgraph'.

- circle(int x, iny y, radius);

  It is used to draw a circle with center (x, y) and the third parameter specifies the radius.

- outtextxy(x,y,"Text");

  It printd text at the (x, y) position.

- rectangle(int left, int top, int right, int bottom);

  It draws a rectangle at the specified coordinates.

- setfillstyle(int pattern, int colour);

  It setd the current fill pattern and fill colour.

<dos.h>

- sound(int frequency);

  It produces a sound of a specific frequency.

- delay(int time);

  It is used to suspend the execution of a program for specific time.

- nosound();

  Turns off the PC speaker.

<conio.h>

- kbhit()

  Determine if the key has been pressed or not, if key has been pressed thet returns a non 0 valueor else returns 0.

A linked list is implemented using the concept of structures and pointers. Self referencial pointer plays a major role in the linking betIen two nodes in a linked list.

Algorithm for creation of a linked list:

1. Create the head node in memory by using the malloc function.

Syntax: headnode=(struct node *)malloc(sizeof(struct node)

2. Populate the head node with data and mark the reference to the next node as NULL. The node pointer should also point to the same node as the headnode.

3. Run a for or while loop with proper conditions to build up nodes.

4. Every iteration involves the creation of a new node by using the malloc function, populating the new node with data, linking this new node to the previous node using the self-referential pointer next and pointing the node pointer to the current node.

Syntax: newnode=(struct node *)malloc(sizeof(struct node)

```
node->next=newnode;

node=newnode;

newnode->next=NULL;
```

5. A linked list is displayed by using a simple while condition and the incrementation of a pointer.

Syntax: while(node->next!=NULL)

```
{
        node=node->next;
            }
```

Tic Tac Toe:

Tic Tac Toe is a very classic game which involves two players 'X' and 'O'. The tactic of the game is that one of the player chooses one symbol and fills up one of the cells of a 3x3 grid. The other player respectively does the same with the other symbol. During the progression of the game, the player must try to make a move such that he has on whole row/column/diagonal filled with his respective characters 'X' or 'O'. If no player wins, then the game is a tie. Various patterns have been developed to master this game.

List of errors-Fixed

Snake game

- Initially, I kept the data for snake and the food under the same structure. Because of that, the 'loc' variable was very difficult to use. That variable was used for deciding the location of snake and the food separately. For solving that error, I created separate structures for food and snake and the problem was solved.

- I Ire able to create the game, but it had no difficulty levels in the beginning. It was obvious to use the switch case for this purpose. In the draw function, I inserted the switch case and hoped it would work, but it didn't. The program shoId error in that function. Then I modified the entire function so that difficulty level concept can be implemented.

- This game is supposed to have food appear at random places each time it was eaten. In the beginning, every time food was eaten, it appeared at the same place again and again. I Ire not able to figure out how to solve this. After reading books from the library, I found 'random()' with which this issue was solved.

```
f->floc.x=150+random(245);
f->floc.y=150+random(245);

f.floc.x=150+random(250);
f.floc.y=150+random(250);
```

- Also, I changed the size of the body of the snake along with its colour per difficulty level so that I could differentiate the head of the snake from its body.

1
1

Tic tac toe

- I used loops for allocating memory space as Ill as for creating links betIen all nodes but the game was not functioning properly as it encountered a runtime error. So, I allocated memory space and creating links for each individual node and the problem was solved.

- Second issue I solved Ire the tie cases. If a triangular placement of the symbol is done by the player, it ends up as a tie. I added those cases for including the tie factor.

- Previously, when number less than 1 or greater than 9 Ire entered, the game shoId error. For fixing that, I put the for loop and fixed the error by skipping the turn of the player entering the wrong values.

1
2

Implementation

Code (Tic Tac Toe)

```
#include<stdio.h>
#include<stdlib.h>

struct Node
{
  char sym;
  int counter;
  struct Node *next;
};

int main()
{
    int a,b,c,z;

  printf("\t\t\t\tTic Tac Toe Game!!!\n\n\n");
  printf("\t\t\t\tCreated by Jeet Bangoria\n\n") ; struct Node*
  head = NULL;
  struct Node* second = NULL;
  struct Node* third = NULL;
  struct Node* fourth = NULL;
  struct Node* fifth = NULL;
```

```
struct Node* sixth = NULL;
struct Node* seventh = NULL;
struct Node* eight = NULL;
struct Node* ninth = NULL;
struct Node* tenth = NULL;

//allocated memory space

head = (struct Node*)malloc(sizeof(struct Node));
second = (struct Node*)malloc(sizeof(struct Node));
third = (struct Node*)malloc(sizeof(struct Node));
fourth = (struct Node*)malloc(sizeof(struct Node));
fifth = (struct Node*)malloc(sizeof(struct Node));
sixth = (struct Node*)malloc(sizeof(struct Node));
seventh = (struct Node*)malloc(sizeof(struct Node));
eight = (struct Node*)malloc(sizeof(struct Node));
ninth = (struct Node*)malloc(sizeof(struct Node));
tenth = (struct Node*)malloc(sizeof(struct Node));

//nodes are allocated to their respective position and createdlinks
betIen all nodes

head->counter = 1;
head->sym=' ';
head->next = second;

second->counter = 2;
```

```
second->sym=' ';
second->next = third;

third->counter = 3;
third->sym=' ';
third->next = fourth;

fourth->counter = 4;
fourth->sym=' ';
fourth->next = fifth;

fifth->counter = 5;
fifth->sym=' ';
fifth->next = sixth;

sixth->counter = 6;
sixth->sym=' ';
sixth->next = seventh;

seventh->counter = 7;
seventh->sym=' ';
seventh->next = eight;

eight->counter = 8;
eight->sym=' ';
eight->next = ninth;
```

```
  ninth->counter = 9;
  ninth->sym=' ';
  ninth->next = tenth;

  tenth->next = NULL;

//displays layout of the game

printf("\t|\t__%d__\t |\t__%d__\t |\t__%d__\t |\n",head->counter,second->counter,third->counter);
printf("\t|\t__%d__\t |\t__%d__\t |\t__%d__\t |\n",fourth->counter,fifth->counter,sixth->counter);
printf("\t|\t__%d__\t |\t__%d__\t |\t__%d__\t |\n",seventh->counter,eight->counter,ninth->counter);


printf("\nPlayer 1(X)\n");
printf("\nPlayer 2(O)\n");
printf("\nSelect your position\n");

//displaying output

void board(){
printf("\t|\t__%c__\t |\t__%c__\t |\t__%c__\t |\n",head->sym,second->sym,third->sym);
printf("\t|\t__%c__\t |\t__%c__\t |\t__%c__\t |\n",fourth->sym,fifth->sym,sixth->sym);
```

```
printf("\t|\t__%c__\t |\t__%c__\t |\t__%c__\t |\n",seventh->sym,eight->sym,ninth->sym);
}

//all the cases for a player to win

void win(){
if( (head->sym=='X') && (second->sym=='X') && (third->sym=='X') ){
   printf("\nPlayer 1 won\n");
   c=10;
                        }
else if( (fourth->sym=='X') && (fifth->sym=='X') && (sixth->sym=='X') ){
   printf("\nPlayer 1 won\n");
   c=10;
                        }
else if( (seventh->sym=='X') && (eight->sym=='X') && (ninth->sym=='X') ){
   printf("\nPlayer 1 won\n");
   c=10;
                        }
else if( (head->sym=='X') && (fourth->sym=='X') && (seventh->sym=='X') ){
   printf("\nPlayer 1 won\n");
   c=10;
                        }
else if( (second->sym=='X') && (fifth->sym=='X') && (eight->sym=='X') ){
   printf("\nPlayer 1 won\n");
   c=10;
                        }
```

```
else if( (third->sym=='X') && (sixth->sym=='X') && (ninth->sym=='X') ){
    printf("\nPlayer 1 won\n");
    c=10;
                        }
else if( (head->sym=='X') && (fifth->sym=='X') && (ninth->sym=='X') ){
    printf("\nPlayer 1 won\n");
    c=10;
                        }
else if( (third->sym=='X') && (fifth->sym=='X') && (seventh->sym=='X') ){
    printf("\nPlayer 1 won\n");
    c=10;
                        }
else if( (head->sym=='O') && (second->sym=='O') && (third->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;
                        }
else if( (fourth->sym=='O') && (fifth->sym=='O') && (sixth->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;
                        }
else if( (seventh->sym=='O') && (eight->sym=='O') && (ninth->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;
                        }
else if( (head->sym=='O') && (fourth->sym=='O') && (seventh->sym=='O') ){
    printf("\nPlayer 2 won\n");
```

```
    c=10;
                        }
  else if( (second->sym=='O') && (fifth->sym=='O') && (eight->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;
                        }
  else if( (third->sym=='O') && (sixth->sym=='O') && (ninth->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;
                        }
  else if( (head->sym=='O') && (fifth->sym=='O') && (ninth->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;
                        }
  else if( (third->sym=='O') && (fifth->sym=='O') && (seventh->sym=='O') ){
    printf("\nPlayer 2 won\n");
    c=10;}

  else if( (third->sym=='O') && (head->sym=='O') && (eight->sym=='O') ){
    printf("\nTie Game!\n");
    c=10;}

  else if( (third->sym=='X') && (head->sym=='X') && (eight->sym=='X') ){
    printf("\nTie Game!\n");
    c=10;}
```

```
else if( (second->sym=='O') && (ninth->sym=='O') && (seventh->sym=='O')
){
   printf("\nTie Game!\n");
   c=10;}

else if( (second->sym=='X') && (ninth->sym=='X') && (seventh->sym=='X') ){
   printf("\nTie Game!\n");
   c=10;}

else if( (head->sym=='O') && (sixth->sym=='O') && (seventh->sym=='O') ){
   printf("\nTie Game!\n");
   c=10;}

else if( (head->sym=='X') && (sixth->sym=='X') && (seventh->sym=='X') ){
   printf("\nTie Game!\n");
   c=10;}

else if( (third->sym=='O') && (fourth->sym=='O') && (ninth->sym=='O') ){
   printf("\nTie Game!\n");
   c=10;}

else if( (third->sym=='X') && (fourth->sym=='X') && (ninth->sym=='X') ){
   printf("\nTie Game!\n");
   c=10;}

}
```

```
c=0;
while(c<9){
 if(c%2==0){
   struct Node *temp=head;
   printf("\nFirst Player's turn:");
   scanf("%d",&a);
   if(a<10&&a>0){
   while(temp->next!=NULL){
      if(temp->counter==a){
         temp->sym='X';
         board();
         win();
         temp=tenth;       // assigning tenth to temp for proper termination of
round
      }
      else{
         temp=temp->next;    //for traversal to next node
      }}
   }else{printf("\nWrong input\nTurn skipped\n");}
   c++;
   }
 else{
   struct Node *temp1=head;
   printf("\nSecond Player's turn:");
   scanf("%d",&b);
   if(b<10&&b>0)
   {
```

```
        while(temp1->next!=NULL){
        if(temp1->counter==b){
           temp1->sym='O';
           board();
           win();
           temp1=tenth;
        }
        else{
           temp1=temp1->next;
        }}
     }
     else{printf("\nWrong input\nTurn skipped\n");}
     c++;
   }
 }
  return 0;
 }
```

Code (Snake Game)

```
//.....FILE INCLUDED.....

#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<alloc.h>
#include<stdlib.h>
#include<dos.h>


//---------------REQUIRED STRUCTURE----------------


char key;

struct loc
{
    int x,y;
};

struct snake
{
    struct loc sloc;
    struct snake *link;
    char dir;
};
```

```
2
3


struct game_data
{
    int score;
    int no_food;

};
struct game_data gd={0,0};




struct limit
{
    int lx1,ly1,lx2,ly2;
};

struct limit l={96,96,404,404};



struct food
{
    struct loc floc;
    int number;
};

int n=0,ch;
```

```
//.....
//.....FUNCTIONS.....
//.....


gameover(void)
{
    cleardevice();
    outtextxy(100,100,"---------- GAME OVER---------- ");

    printf("\n\n\n\n\n\n\n\n\n");
    printf("\t\t# Score      - %d",gd.score);
    printf("\n\t\t# Food eaten - %d", gd.no_food);
    outtextxy(200,300,"Press any key to exit");
    s1:
    sound(300);delay(300);sound(450);delay(150);sound(500);delay(150);
    sound(300);delay(200);sound(450);delay(100);sound(450);delay(200);
    while(!kbhit()){goto s1;}
    nosound();

     exit(0);
    return 0;
}
draw(struct snake *head,struct food *f,int ch)
{
```

```
struct snake *temp;
temp=head;

switch(ch)
{
        case 1:

                rectangle(96,96,404,404);
                rectangle(98,98,402,402);
                rectangle(100,100,400,400);
                setfillstyle(9,13);
                outtextxy(220,40,"Easy");
                bar(temp->sloc.x-6,temp->sloc.y-6,temp->sloc.x+6,temp->sloc.y+6);
                temp=temp->link;
                setfillstyle(9,2);
                while(temp->link!=NULL)
                {

                        bar(temp->sloc.x-5,temp->sloc.y-5,temp->sloc.x+5,temp->sloc.y+5);
                        temp=temp->link;
                }

                bar(temp->sloc.x-5,temp->sloc.y-5,temp->sloc.x+5,temp->sloc.y+5);
```

```
                circle(f->floc.x-2,f->floc.y-2,3);
                circle(f->floc.x+2,f->floc.y+2,3);
                circle(f->floc.x-2,f->floc.y+2,3);
                circle(f->floc.x+2,f->floc.y-2,3);

                delay(25);
                while(!kbhit()){goto e;}
                key=getche();

        break;

    case 2:

                rectangle(96,96,404,404);
                rectangle(98,98,402,402);
                rectangle(100,100,400,400);
                setfillstyle(9,13);
                outtextxy(220,40,"Medium");
                bar(temp->sloc.x-6,temp->sloc.y-6,temp->sloc.x+6,temp-
>sloc.y+6);
                temp=temp->link;
                setfillstyle(9,BLUE);
                while(temp->link!=NULL)
                {

                        bar(temp->sloc.x-5,temp->sloc.y-5,temp-
>sloc.x+5,temp->sloc.y+5);
```

```
                temp=temp->link;
            }

            bar(temp->sloc.x-5,temp->sloc.y-5,temp->sloc.x+5,temp-
>sloc.y+5);

            circle(f->floc.x-2,f->floc.y-2,3);
            circle(f->floc.x+2,f->floc.y+2,3);
            circle(f->floc.x-2,f->floc.y+2,3);
            circle(f->floc.x+2,f->floc.y-2,3);

            delay(20);
            while(!kbhit()){goto e;}
            key=getche();

        break;

    case 3:

            rectangle(96,96,404,404);
            rectangle(98,98,402,402);
            rectangle(100,100,400,400);
            setfillstyle(9,13);
            outtextxy(220,40,"Hard");
            bar(temp->sloc.x-6,temp->sloc.y-6,temp->sloc.x+6,temp-
>sloc.y+6);
            temp=temp->link;
```

```
                setfillstyle(9,RED);
                while(temp->link!=NULL)
                {

                        bar(temp->sloc.x-5,temp->sloc.y-5,temp->sloc.x+5,temp->sloc.y+5);
                        temp=temp->link;
                }

                bar(temp->sloc.x-5,temp->sloc.y-5,temp->sloc.x+5,temp->sloc.y+5);

                circle(f->floc.x-2,f->floc.y-2,3);
                circle(f->floc.x+2,f->floc.y+2,3);
                circle(f->floc.x-2,f->floc.y+2,3);
                circle(f->floc.x+2,f->floc.y-2,3);

                delay(10);
                while(!kbhit()){goto e;}
                key=getche();
            break;
    }
    e:
     cleardevice();
     return 0;
}
void game(struct snake *head,struct food *f,int ch)
```

```
{
    struct snake *temp,pre,nxt;
    temp=head;

    while(key!='p')
    {
    if(head->sloc.x==l.lx1||head->sloc.x==l.ly2||head->sloc.y==l.ly1||head->sloc.y==l.ly2)
    {gameover();}
    if(head->sloc.x>=f->floc.x-5&&head->sloc.x<=f->floc.x+5&&head->sloc.y>=f->floc.y-5&&head->sloc.y<=f->floc.y+5)
    {
            temp=head;
            sound(420);

            f->floc.x=150+random(245);
            f->floc.y=150+random(245);
            gd.score+=100;
            gd.no_food+=1;
            n=n+1;

            while(temp->link!=NULL){temp=temp->link;}
            temp->link=(struct snake *)malloc(sizeof(struct snake));
            temp->link->link=NULL;
            temp->link->sloc.x= temp->sloc.x;
            temp->link->sloc.y= temp->sloc.y;
```

```
            temp->link->dir=temp->dir;
            n=0;
          }
        switch(key)
        {
         case 'a': if(head->dir!='d'){head->dir='a'; head->sloc.x-=2; } else
{key=head->dir;} break;
         case 'w': if(head->dir!='s'){head->dir='w'; head->sloc.y-=2; } else
{key=head->dir;} break;
         case 'd': if(head->dir!='a'){head->dir='d'; head->sloc.x+=2; } else
{key=head->dir;} break;
         case 's': if(head->dir!='w'){head->dir='s'; head->sloc.y+=2; } else
{key=head->dir;} break;

        }

         draw(head,f,ch);
         nosound();
         temp=head;
         pre=*temp;

         while(temp->link!=NULL)
        {
            nxt.sloc.x=temp->link->sloc.x;
            nxt.sloc.y=temp->link->sloc.y;
            nxt.dir=temp->link->dir;
            temp->link->sloc.x=pre.sloc.x;
            temp->link->sloc.y=pre.sloc.y;
```

```
                temp->link->dir=pre.dir;
                temp=temp->link;
                pre=nxt;
        }


    }

}


void main(void)
{

    struct snake *s;
    struct food f;
    int gdriver = DETECT, gmode, errorcode;



  clrscr();

  s=(struct snake *)malloc(sizeof(struct snake));
  s->dir='w';
  s->link=NULL;
  s->sloc.x= s->sloc.y=300;
```

```
f.floc.x=150+random(250);
f.floc.y=150+random(250);
//.....

initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

//-------- # ENTER REQUIRED PATH FOR EGAVGA.BGI----------

outtextxy(300,50,"The SNAKE Game!!!");
outtextxy(220,100,"Created by Darsh, Nimit and Alekhya");
outtextxy(100,160,"# Instructions :");
outtextxy(100,200,"1. Press 'w' to go upward");
outtextxy(100,220,"2. Press 's' to go downward");
outtextxy(100,240,"3. Press 'a' to go left hand side");
outtextxy(100,260,"4. Press 'd' to go right hand side");
outtextxy(100,280,"5. Press 'p' to pause the game");
outtextxy(100,320,"CHOOSE DIFFULCITY:");
outtextxy(100,340,"1. Easy");
outtextxy(100,360,"2. Medium");
outtextxy(100,380,"3. Hard");
scanf("%d",&ch);
s:
sound(700);delay(200);sound(450);delay(180);sound(300);delay(150);
sound(350);delay(200);sound(450);delay(180);sound(500);delay(150);
while(!kbhit()){goto s;}
```

```
3
3

    nosound();



    pause :
    draw(s,&f,ch);
    game(s,&f,ch);
    key='r';
    rectangle(60,60,600,200);
    outtextxy(260,100,"GAME PAUSED");
    s1:
    sound(400);delay(500);sound(450);delay(250);sound(300);delay(200);
    sound(400);delay(300);sound(450);delay(250);sound(300);delay(200);
    while(!kbhit()){goto s1;}
    goto pause;



    closegraph();
    clrscr();

    printf("\n\n\n\n\t\t\t\tThankyou.....");
    getch();}
```

OUTPUT:

CONCLUSION:

- As I can see from the above code of snake game, I are able to maintain a link betIen current position and previous position of snake. This is very much applicable in the real world too as it can be used in the undo functionality in word.
- Link list can be used to solve a large number of polynomial problems by simply storing the coefficient and exponent of each term.
- Link list are often used in bioinformatics applications, searching for patterns in DNA or protein sequences. The ability to search efficiently with mismatches might be considered their greatest strength.
- Circular link list are used in our computers and mobiles where multiple applications are running. A link list is also used in implementation of stacks, queues, graphs, creating a hash table.

REFERENCES

- Wikipedia
- YouTube

BOOKS REFFERED

- Mastering in C –by Venugopal
- Let us C –by Yashvant Kanetkar