lab4

Bhupinder Singh

March 25, 2019

Contents

1 lab4.cpp

General Description - For this lab, we will be working with linked lists
to store polynomials.
The general topic of Data Structures consists of multiple things for example,
algorithms and a LOT of data. In order to accurately store and use this data,
we need different methods to traverse through this data. This lab introduces
us to the concept of linked lists. A linked list is a linear data structure
where each element is a separate object. Each element (we will call it a node)
of a list is comprising of two items - the data and a reference to the next node.
The last node has a reference to null. The entry point into a linked list is called
the head of the list. Using linked lists is a very useful way to traverse through
large sets of data. The purpose of this lab is to get us familiar with these linked
lists and be efficient in using them. Also, the usage of linked list is required for
this lab.
This program uses the cairo 2d graphics library within an FLTK window to
display a polynomial: e.g.

$$6x^2 + 9x + 8$$

```
#include <iostream>
#include "config.h"
#include <FL/Fl_Value_Input.H>
#include <FL/Fl_Output.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Cairo_Window.H>
#include <string>

using namespace std;


structure that stores the node.
struct node
{
        int cof, exp;
        node *next;
};

function declarations
void drawCallBack(Fl_Cairo_Window *, cairo_t *cr);
void display(node *head);
void termInput1(Fl_Widget*, void *);
void termInput2(Fl_Widget*, void *);
void termRemove1(Fl_Widget*, void *);
void termRemove2(Fl_Widget*, void *);
void addition(Fl_Widget*, void *);
void multiplication(Fl_Widget*, void *);
void insert(int cof, int exp, node *&head);
void remove(node *&head);
void remove_everything(node *&head);
```

```
required class polynomial that stores the polynomial itself
class Polynomial
{
        public:
                Polynomial() { first = NULL; second = NULL; }
                ~Polynomial() {
                        remove_everything(first); remove_everything(second);
                        remove_everything(add); remove_everything(mult);
                }
                node *first;
                node *second;
                node *add;
                node *mult;
};

string Exponent[10] = {
        u8"", u8"x", u8"x\u00B2", u8"x\u00B3", u8"x\u2074", u8"x\u2075",
        u8"x\u2076", u8"x\u2077", u8"x\u2078", u8"x\u2079"
};


const int Width = 400;
const int Height = 400;
Polynomial Poly;
string output;
Fl_Value_Input *cofInput;
Fl_Value_Input *expInput;
Fl_Cairo_Window pw(Width, Height);

int main()
{
        Fl_Cairo_Window iw(Width, Height);
```

```
        Fl_Output x(5*Width/8, 5*Height/16, 0, 0, "X");
        x.labelsize(40);
        cofInput = new Fl_Value_Input(Width/4, Height/4, Width/4, Height/8,"
            Coeficient: ");
        expInput = new Fl_Value_Input(3*Width/4, Height/8, Width/8, Height
            /8,"Exponent: ");

        Fl_Button term1(Width/4, 3*Height/4, Width/4, Height/8,"Enter Term 1
            ");
        Fl_Button term2(Width/2, 3*Height/4, Width/4, Height/8,"Enter Term 2
            ");
        Fl_Button remove1(Width/4, 3.5*Height/4, Width/4, Height/8,"Enter
            Term 1");
        Fl_Button remove2(Width/2, 3.5*Height/4, Width/4, Height/8,"Enter
            Term 2");
        Fl_Button resAdd(3*Width/4, 3*Height/4, Width/4, Height/8,"Addition"
            );
        Fl_Button resMult(3*Width/4, 3.5*Height/4, Width/4, Height/8,"
            Multiplication");

        term1.callback(termInput1);
        term2.callback(termInput2);
        remove1.callback(termRemove1);
        remove2.callback(termRemove2);
        resAdd.callback(addition);
        resMult.callback(multiplication);
        iw.show();
        iw.end();
        pw.set_draw_cb(drawCallBack);
        //pw.show();
        return Fl::run();
}
```

```
void drawCallBack(Fl_Cairo_Window *, cairo_t *cr)
{
        cairo_set_source_rgb(cr, 1, 0, 0);      //red
        cairo_move_to(cr, Width/2, Height/2);   //draw to middle of screen
        cairo_set_font_size(cr, 24);
        cairo_show_text(cr, output.c_str());
//      cairo_show_glyph()
}

Function that removes data and clears up the memory.
void remove(node *&head)
{
        node *temp;

        temp = head;
        head = head->next;
        if (temp)
                delete temp;
}
Function that removes data and clears up the memory.
void remove_everything(node *&head)
{
        node *temp;

        while (head)
        {
                temp = head;
                head = head->next;
                delete temp;
        }
}
```

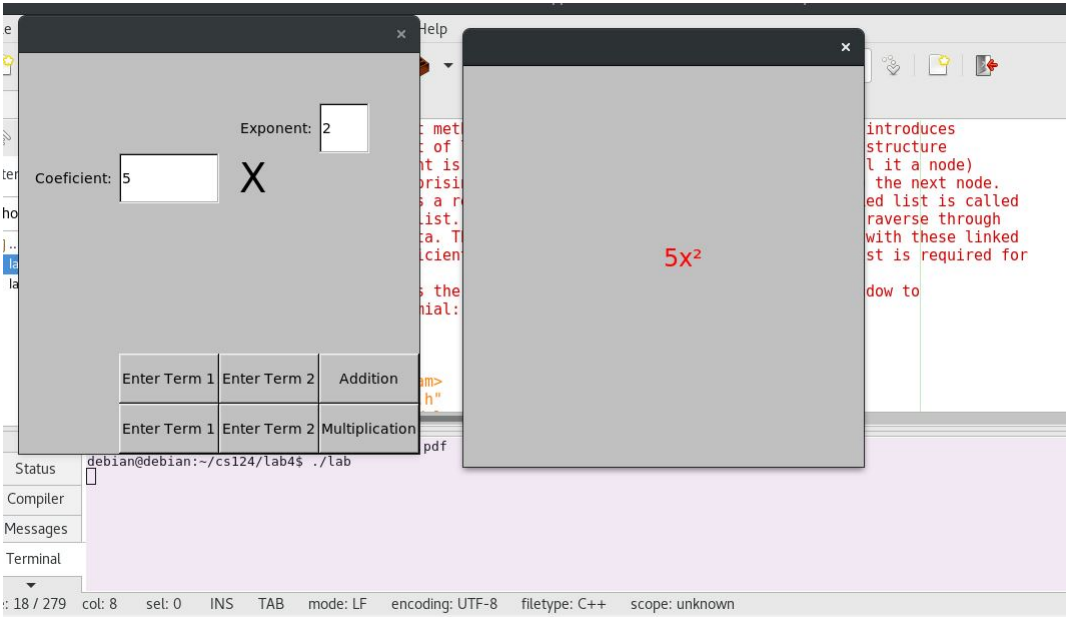
```
The following function traverses through the lists and displays the
content inside the lists.
void display(node *head)
{
        node *tmp;

        tmp = head;
        output.clear();
        while (tmp)
        {
                output += to_string(tmp->cof);
                if (0 <= tmp->exp && tmp->exp <= 9)
                        output += Exponent[tmp->exp] + " ";
                else
                        output += "x^" + to_string(tmp->exp);
                tmp = tmp->next;
                if (tmp)
                        output += " + ";
        }
}
function that takes in the first term value
void termInput1(Fl_Widget*, void *)
{
        insert(cofInput->value(), expInput->value(), Poly.first);
        display(Poly.first);
        pw.show();
}
function that takes in the second term value
void termInput2(Fl_Widget*, void *)
{
        insert(cofInput->value(), expInput->value(), Poly.second);
```

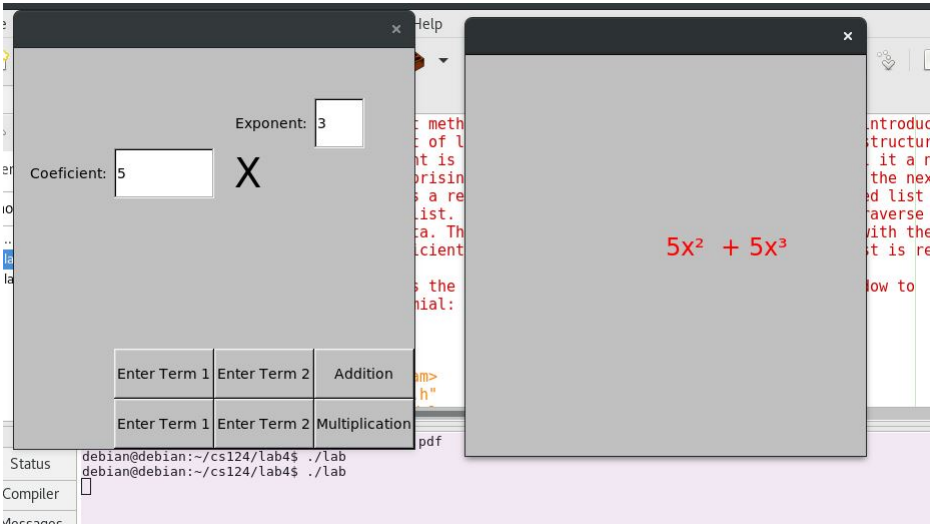
```
        display(Poly.second);
        pw.show();
}
function that removes the first term value to save memory
void termRemove1(Fl_Widget*, void *)
{
        remove(Poly.first);
        display(Poly.first);
        pw.show();
}
function that removes the second term value to save memory
void termRemove2(Fl_Widget*, void *)
{
        remove(Poly.second);
        display(Poly.second);
        pw.show();
}
Following function adds the given values inputted by the user.
The function does that by traversing through the lists while its not empty
and inserting the given values.
void addition(Fl_Widget*, void *)
{
        node *tmp;
        tmp = Poly.first;
        while (tmp)
        {
                insert(tmp->cof, tmp->exp, Poly.add);
                tmp = tmp->next;
        }
        tmp = Poly.second;
        while (tmp)
        {
```

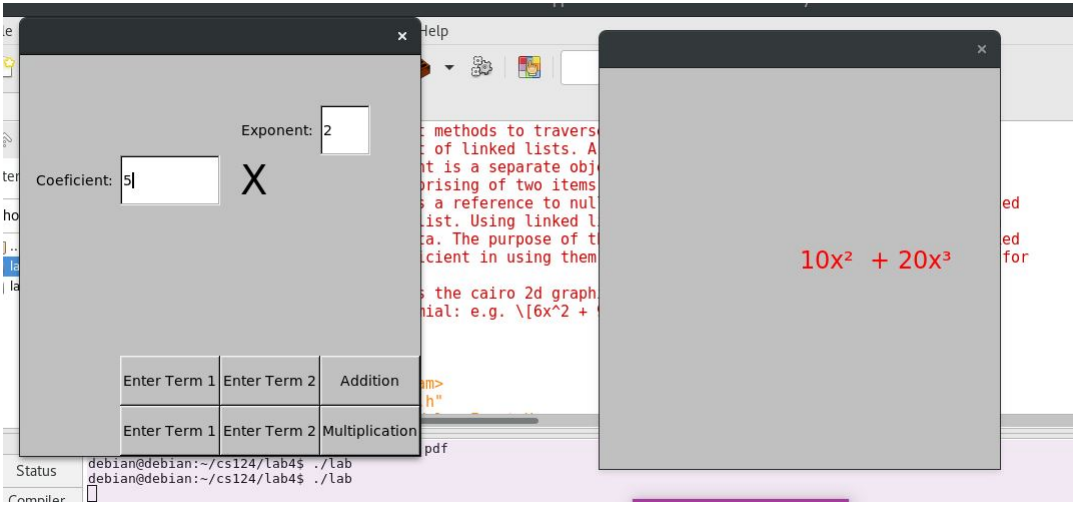
```
                insert(tmp->cof, tmp->exp, Poly.add);
                tmp = tmp->next;
        }
        display(Poly.add);
        pw.show();
}
Following function multiplies the given values inputted by the user.
The function does that by traversing through the lists while its not empty
and inserting the given values.
void multiplication(Fl_Widget*, void *)
{
        node *t1, *t2;

        t1 = Poly.first;
        while (t1)
        {
                t2 = Poly.second;
                while (t2) //while true
                {
                        insert(t1->cof * t2->cof, t1->exp * t2->exp, Poly.
                            mult); //multiply
                        t2 = t2->next; //traverses as long as t2 is true
                }
                t1 = t1->next;
        }
        display(Poly.mult); //display
        pw.show();
}
the backbone function of the whole code, this function inserts the given
parameters into the linked lists.
void insert(int cof, int exp, node *&head)
{
```

```
node *newnode = new node;

newnode->cof = cof;
newnode->exp = exp;
newnode->next = NULL;
if(head == NULL)
        head = newnode;
else
{
        node *curr = head;
        node *traverse = NULL;
        while (curr != NULL)
        {
                if (curr->exp == newnode->exp) //compares current
                    exponent to next
                {
                        curr->cof += newnode->cof; //adds the cof if
                             they are same
                        delete newnode;
                        return ;
                }
                if (curr->exp > newnode->exp)//breaks if the curr is
                     more than next
                        break;
                else
                {
                        traverse = curr;
                        curr = curr->next;
                }
        }
        if (curr == head)
        {
```

```
                        newnode->next = head;
                        head = newnode;
                }
                else
                {
                        newnode->next = curr;
                        traverse->next = newnode;
                }
        }
}
```


Exponent: 2
Coeficient: 5
X
Enter Term 1
Enter Term 2
Addition
Enter Term 1
Enter Term 2
Multiplication
5x²
debian@debian:~/cs124/lab4$ ./lab
Status
Compiler
Messages
Terminal
: 18 / 279 col: 8 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: C++ scope: unknown


Help
Exponent: 3
Coeficient: 5
X
Enter Term 1
Enter Term 2
Addition
Enter Term 1
Enter Term 2
Multiplication
5x² + 5x³
pdf
Status
Compiler
debian@debian:~/cs124/lab4$ ./lab
debian@debian:~/cs124/lab4$ ./lab


Exponent: 2
Coeficient: 5
X
Enter Term 1
Enter Term 2
Addition
Enter Term 1
Enter Term 2
Multiplication
10x² + 20x³
Status
debian@debian:~/cs124/lab4$ ./lab
debian@debian:~/cs124/lab4$ ./lab

For the record, I would just like to say that this lab was very hard for me, I had a recently surgery and had to miss class three times. Even though I missed class, my friends and the tutor Mingyun Kim helped me out a lot. I am glad that I was able to complete this much of the assignment thanks to them.