1. **Program Statement**

This program takes an array of elements with template data type and uses the array to create a stack of ascending linked list. User can check the minimum and maximum values that would be determined by writing a function and introduce the idea of pushing, popping, peeking, checking if the stack is full, or checking if the stack is empty.

1. **Requirements**
   1. **Assumptions**
      1. The operators < and > are defined for the class T
      2. User types in the same data type for each class A and B
         1. Float, char, or int
      3. Inputs are all different than one another
      4. Number of data in each array is 5
      5. Input is either true or false for minimum value
   2. **Specifications**
      1. Input output files
         1. Input file is filled with data already
      2. Do tests using int and float
      3. Perform test to see if the files can be found or not
      4. A function that takes in datatype template T and Boolean min and returns the minimum or maximum value
2. **Decomposition Diagram**

|  |  |  |
| --- | --- | --- |
| **Main** | | |
| **Input** | **Process** | **Output** |
| Integer numbers | Call the function to find the minimum and maximum values after sorting it using linked lists as stacks | Output the minimum and maximum values in organized way |
| Float numbers | Call the function to find the minimum and maximum values after sorting it using linked lists as stacks | Output the minimum and maximum values in organized way |

1. **Test Strategy**
   1. **Valid Data**
   2. **Invalid Data**
2. **Test Plan Version 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | 5 positive integer values that are different than one another |  |  |  |  |
| Valid | 2 | 5 positive float values that are different that on another |  |  |  |  |

1. **Initial Algorithm**
   1. In main
      1. Using while loop, check to see if the file given opens or not
         1. If it doesn’t open, then print an error message
      2. Using the same while loop, fill up two separate arrays of int and float with datas from the file
      3. Call the function to find min and max values and print them to screen and an output file
   2. In function
      1. Make 3 nodes of linked list and set two of them to NULL
      2. Make a new list
      3. Then if user passes in false for min
         1. Allocate a new memory, set data of a node to what is in array, and while a counter is less than 4, check if that array data is less than the last stored value in the array
         2. If yes, call the function to pop.
         3. continue looping until the end of the array
      4. Continue doing these tests until the end of the linked list
      5. Return the first value in the head node, if user wants minimum, else return the last value, which is the maximum
2. **Test Plan Version 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | 5 positive integer values that are different than one another | 4, 1, 13, 3, 2 | Min: 1  Max: 13 |  |  |
| Valid | 2 | 5 positive float values that are different that on another | 1.1, 4.1, 8.1, 5.2, 2.3 | Min: 1.1  Max: 8.1 |  |  |
| Invalid | 1 | File cannot be found | File name: something.txt | File cannot be found |  |  |

1. **Code**

/\*

Program: Lab 9.1

Author: Nisarg Patel

Creation Date: 4/4/18

Modification Date: 4/6/2018

\*/

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

template <typename T>

T minMaxFunc(T arrays[], bool min) {

struct list {

T data;

list \*next;

};

list \*temp = new list;

temp->data = 0;

temp->next = NULL;

list \*head = NULL;

list \*curr = NULL;

head = temp;

curr = head;

int i = 0, x;

T tempArray[5];

if (min == false) {

i = 0;

temp = new list;

temp->data = arrays[i];

temp->next = NULL;

curr = temp;

head->next = curr;

i++;

while (i <= 4) {

if (arrays[i] < curr->data) {

x = 0;

while (arrays[i] < curr->data) {

temp = pop(head);

tempArray[x] = temp->data;

if (curr->next == NULL) {

curr = head;

head->next = NULL;

break;

}

else {

head->next = curr->next;

curr = curr->next;

}

x++;

}

temp = new list;

temp->data = arrays[i];

temp->next = NULL;

curr->next = temp;

curr = temp;

i++;

for (int i = 0; i < x; i++) {

temp = new list;

temp->data = tempArray[i];

temp->next = curr;

head->next = temp;

curr->next = NULL;

curr = temp;

}

}

else {

temp = new list;

temp->data = arrays[i];

temp->next = curr;

head->next = temp;

curr->next = NULL;

curr = temp;

i++;

}

}

}

else {

i = 0;

temp = new list;

temp->data = arrays[i];

temp->next = NULL;

curr = temp;

head->next = curr;

i++;

while (i <= 4) {

if (arrays[i] > curr->data) {

x = 0;

while (arrays[i] > curr->data) {

temp = pop(head);

tempArray[x] = temp->data;

if (curr->next == NULL) {

curr = head;

head->next = NULL;

x++;

break;

}

else {

head->next = curr->next;

curr = curr->next;

}

x++;

}

temp = new list;

temp->data = arrays[i];

temp->next = NULL;

head->next = temp;

curr = temp;

i++;

for (int i = x - 1; i >= 0; i--) {

temp = new list;

temp->data = tempArray[i];

temp->next = curr;

head->next = temp;

curr->next = NULL;

curr = temp;

}

}

else {

temp = new list;

temp->data = arrays[i];

temp->next = curr;

head->next = temp;

curr->next = NULL;

curr= temp;

i++;

}

}

}

T result = curr->data;

return result;

}

template <typename T>

T pop(T thisPop) {

T result = thisPop->next;

return result;

}

int main()

{

ifstream fin1, fin2;

fstream fout;

int x, i = 0;

float y;

int arrayInt[5];

float arrayFloat[5];

fin1.open("data.dat");

if (!fin1) {

cout << "Cannot find the file" << endl;

}

while (fin1 >> x) {

arrayInt[i] = x;

i++;

}

fin1.close();

i = 0;

fin2.open("data.dat");

if (!fin2) {

cout << "Cannot find the file" << endl;

}

while (fin2 >> y) {

arrayFloat[i] = y;

i++;

}

fin2.close();

fin2.open("out.txt");

cout << "Row 1 (int)" << endl;

cout << "Max Value: " << minMaxFunc<int>(arrayInt, false) << endl;

cout << "Min Value: " << minMaxFunc<int>(arrayInt, true) << endl << endl;

fout << "Row 1 (int)" << endl;

fout << "Max Value: " << minMaxFunc<int>(arrayInt, false) << endl;

fout << "Min Value: " << minMaxFunc<int>(arrayInt, true) << endl << endl;

cout << "Row 2 (float)" << endl;

cout << "Max Value: " << minMaxFunc<float>(arrayFloat, false) << endl;

cout << "Min Value: " << minMaxFunc<float>(arrayFloat, true) << endl << endl;

fout << "Row 2 (float)" << endl;

fout << "Max Value: " << minMaxFunc<float>(arrayFloat, false) << endl;

fout << "Min Value: " << minMaxFunc<float>(arrayFloat, true) << endl << endl;

fout.close();

system("pause");

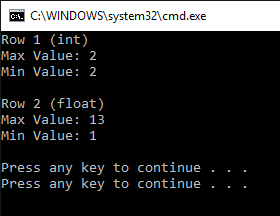
return 0;

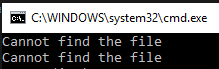
}

1. **Updated Algorithm**
   1. In main
      1. Using while loop, check to see if the file given opens or not
         1. If it doesn’t open, then print an error message
      2. Using the same while loop, fill up two separate arrays of int and float with datas from the file
      3. Call the function to find min and max values and print them to screen and an output file
   2. In function
      1. Make struct with template T data and a pointer list called next to be used for linked list
      2. Make 3 nodes of linked list and set two of them to NULL
      3. Make a new list
      4. Then if user passes in false for min
         1. Allocate a new memory, set data of a node to what is in array, and while a counter is less than 4, check if that array data is less than the last stored value in the array
         2. If yes, call the function to pop.
         3. continue looping until the end of the array
      5. Continue doing these tests until the end of the linked list
      6. Return the first value in the head node, if user wants minimum, else return the last value, which is the maximum
   3. Make a template T right before the function
2. **Test Plan Version 3**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Strategy | Test Number | Description | Input | Expected Output | Actual Output | Pass/Fail |
| Valid | 1 | 5 positive integer values that are different than one another | 4, 1, 13, 3, 2 | Min: 1  Max: 13 | Max: 2  Min: 2 | Fail |
| Valid | 2 | 5 positive float values that are different that on another | 1.1, 4.1, 8.1, 5.2, 2.3 | Min: 1.1  Max: 8.1 | Max: 13  Min 1 | Fail |
| Invalid | 1 | File cannot be found | File name: something.txt | File cannot be found | File cannot be found | Pass |

1. **Screenshots**

 **Valid Test Case 1 And 2**

 **Invalid test Case 1**

1. **Status**

Code does not work correctly as it does not properly store in values, and instead it used those values for float and prints it out for float