Josephus Problem

**JosephusProblem.h**

|  |
| --- |
| // |
|  | // Created by Cristina Lawson on 2019-04-07. |
|  | // |
|  |  |
|  | #include <iostream> |
|  | using namespace std; |
|  |  |
|  | #ifndef THE\_JOSEPHUS\_PROBLEM\_JOSEPHUSPROBLEM\_H |
|  | #define THE\_JOSEPHUS\_PROBLEM\_JOSEPHUSPROBLEM\_H |
|  |  |
|  | struct Node { |
|  | string data; |
|  | Node\* prev; |
|  | Node \*next; |
|  | Node(string data) : data(data), prev(0), next(0) {} |
|  | }; |
|  |  |
|  | class JosephusProblem { |
|  | private: |
|  | Node\* head; |
|  | Node\* tail; |
|  | Node\* curr; |
|  | Node\* off; |
|  | int n; //size |
|  | int pos; //position |
|  | char direction; |
|  | int k; //skipped |
|  |  |
|  | public: |
|  | JosephusProblem(); |
|  | void make(); |
|  | void play(); |
|  | void printCircle(); |
|  | void printSurvivor() const; |
|  | void push\_back(string); |
|  | bool empty() const; |
|  | void setSize(int); |
|  | void setPosition(int); |
|  | void setDirection(char); |
|  | void setSkipped(int); |
|  | void remove(); |
|  |  |
|  | }; |
|  |  |
|  |  |
|  | #endif //THE\_JOSEPHUS\_PROBLEM\_JOSEPHUSPROBLEM\_H |

.CPP

|  |
| --- |
| // |
|  | // Created by Cristina Lawson on 2019-04-07. |
|  | // |
|  |  |
|  | #include <iostream> |
|  | #include <fstream> |
|  | using namespace std; |
|  |  |
|  | #include "JosephusProblem.h" |
|  |  |
|  | JosephusProblem::JosephusProblem() { |
|  | head = nullptr; |
|  | tail = nullptr; |
|  | curr = nullptr; |
|  | off = nullptr; |
|  | } |
|  |  |
|  | void JosephusProblem::setSize(int size) { |
|  | n = size; |
|  | } |
|  |  |
|  | void JosephusProblem::setPosition(int position) { |
|  | pos = position; |
|  | } |
|  |  |
|  | void JosephusProblem::setDirection(char direct) { |
|  | direction = direct; |
|  | } |
|  |  |
|  | void JosephusProblem::setSkipped(int skip) { |
|  | k = skip; |
|  | } |
|  |  |
|  | void JosephusProblem::make() { |
|  | ifstream in; |
|  | string name; |
|  |  |
|  | cout << "Opening file names.txt." << endl; |
|  |  |
|  | in.open("names.txt"); |
|  |  |
|  | if (!in.is\_open()) { |
|  | cout << "Could not open file names.txt." << endl; |
|  | exit(1); |
|  | } |
|  |  |
|  | for (unsigned i = 0; i < n; ++i) { |
|  | in >> name; |
|  | push\_back(name); |
|  | } |
|  |  |
|  | in.close(); |
|  | } |
|  |  |
|  | void JosephusProblem::play() { |
|  |  |
|  | cout << endl << "Making Josephus Problem of Size " << n << "." << endl; |
|  |  |
|  | if (direction == 'R') { |
|  | curr = head; |
|  | if (pos > 1) { |
|  | for (unsigned i = 1; i < pos; ++i) { |
|  | if (curr->next != head) { |
|  | curr = curr->next; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | else if (direction == 'L') { |
|  | curr = tail; |
|  | if (pos > 1) { |
|  | for (unsigned j = 1; j < pos; ++j) { |
|  | if (curr->prev != tail) { |
|  | curr = curr->prev; |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | cout << endl << "Current Josephus Problem: "; |
|  | printCircle(); |
|  |  |
|  | if (direction == 'R') { |
|  | while (curr->next != curr) { |
|  | if (k == 0) { |
|  |  |
|  | } |
|  | else { |
|  | for (unsigned i = 0; i < k; ++i) { |
|  | curr = curr->next; |
|  | } |
|  | } |
|  |  |
|  | remove(); |
|  |  |
|  | --n; |
|  | } |
|  | } |
|  | else if (direction == 'L') { |
|  | while (curr->prev != curr) { |
|  | if (k == 0) { |
|  |  |
|  | } |
|  | else { |
|  | for (unsigned i = 0; i < k; ++i) { |
|  | curr = curr->prev; |
|  | } |
|  | } |
|  |  |
|  | remove(); |
|  |  |
|  | --n; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void JosephusProblem::remove() { |
|  | if (off == head) { |
|  | Node\* temp = head->next; |
|  |  |
|  | if (!empty()) { |
|  | head->next = temp->next; |
|  | temp->next->prev = head; |
|  | delete temp; |
|  | } |
|  | } |
|  | else if (off == tail) { |
|  | Node\* prevNode = tail->prev; |
|  |  |
|  | if (!empty()) { |
|  | tail->prev = prevNode->prev; |
|  | prevNode->prev->next = tail; |
|  | delete prevNode; |
|  | } |
|  | } |
|  | else if (direction == 'R') { |
|  | Node\* temp = nullptr; |
|  | while (curr->next != curr) { |
|  | off = curr->next; |
|  | curr->next = off->next; |
|  | off->next->prev = curr; |
|  | curr = off->next; |
|  | delete off; |
|  | } |
|  | } |
|  | else if (direction == 'L') { |
|  | while (curr->prev != curr) { |
|  | off = curr->prev; |
|  | curr->prev = off->prev; |
|  | off->prev->next = curr; |
|  | curr = off->prev; |
|  | delete off; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void JosephusProblem::printCircle() { |
|  | Node\* i = nullptr; |
|  |  |
|  | cout << head->data; |
|  | cout << " "; |
|  |  |
|  | for (i = head->next; i != head; i = i->next) { |
|  | cout << i->data; |
|  | if (i != tail) { |
|  | cout << " "; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void JosephusProblem::printSurvivor() const { |
|  | cout << curr->data; |
|  | } |
|  |  |
|  | void JosephusProblem::push\_back(string name) { |
|  | Node\* temp = new Node(name); |
|  |  |
|  | if (empty()) { |
|  | head = temp; |
|  | tail = temp; |
|  | head->next = tail; |
|  | tail->next = head; |
|  | head->prev = tail; |
|  | tail->prev = head; |
|  | } |
|  | else { |
|  | temp->next = head; |
|  | head->prev = temp; |
|  | temp->prev = tail; |
|  | tail->next = temp; |
|  | tail = temp; |
|  | } |
|  | } |
|  |  |
|  | bool JosephusProblem::empty() const { |
|  | if (head == nullptr) { |
|  | return true; |
|  | } |
|  | else { |
|  | return false; |
|  | } |
|  | } |

MAIN.CPP

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  |  |
|  | #include "JosephusProblem.h" |
|  |  |
|  | int main() { |
|  | int n = 0; //size |
|  | int pos = 0; |
|  | char direction = ' '; |
|  | int k; //skipped |
|  | JosephusProblem problem; |
|  |  |
|  | cout << "Enter the size (1 - 32): " << endl; |
|  | cin >> n; |
|  | while (n > 32) { |
|  | cout << "Enter the size (1 - 32): " << endl; |
|  | cin >> n; |
|  | } |
|  | problem.setSize(n); |
|  |  |
|  | cout << "Enter the starting position (1 - " << n << "): " << endl; |
|  | cin >> pos; |
|  | while (pos > n || pos < 1) { |
|  | cout << "Enter the starting position (1 - " << n << "): " << endl; |
|  | cin >> pos; |
|  | } |
|  | problem.setPosition(pos); |
|  |  |
|  | cout << "Enter the direction (L or R): " << endl; |
|  | cin >> direction; |
|  | while (!(direction == 'L' || direction == 'R')) { |
|  | cout << "Enter the direction (L or R): " << endl; |
|  | cin >> direction; |
|  | } |
|  | problem.setDirection(direction); |
|  |  |
|  | cout << "Enter number to be skipped (0 - " << n << "): " << endl; |
|  | cin >> k; |
|  | while (k > n || k < 0) { |
|  | cout << "Enter number to be skipped (0 - " << n << "): " << endl; |
|  | cin >> k; |
|  | } |
|  | problem.setSkipped(k); |
|  |  |
|  | problem.make(); |
|  |  |
|  | problem.play(); |
|  |  |
|  | cout << endl << endl << "Survivor: "; |
|  | problem.printSurvivor(); |
|  | cout << endl; |
|  |  |
|  | return 0; |
|  | } |