Міністерство освіти і науки України

Національний Технічний Університет

«Харківський Політехнічний Інститут»

Кафедра «Стратегічного Керування»

Лабораторна робота № 1

«БАЗОВІ СТРУКТУРИ ДАНИХ»

Перевірила: Мошко Є. О.

Виконав:

Харків, 2018

Лабораторна робота № 1

Тема: Базові Структури Даних

Завдання

Розробити програму, яка читає з клавіатури послідовність N цілих чисел (1 < N < 256), жодне з яких не повторюється, зберігає їх до структури даних (згідно завданню) та видає на екран наступні характеристики:

 кількість елементів;

 середнє арифметичне збережених елементів;

 мінімальний та максимальний елемент;

 четвертий елемент послідовності;

 елемент, що йде перед мінімальним елементом.

Наголосимо, що всі характеристики потрібно визначити із заповненої структури даних. Дозволено використовувати лише ті операції, що притаманні заданій структурі, наприклад, заборонено отримувати доступ до елементу із довільною позицією у черзі, яку реалізовано на базі масиву.

Використовувати готові реалізації структур даних (наприклад, STL) заборонено.

Використати наступні структури даних: Стек (другий варіант).

Хід роботи

|  |
| --- |
| #include <iostream>  using namespace std;  struct List {  int data;  int counter;  List\* next;  List\* prev;  };  class Stack {  private:  List\* first = 0;  List\* last = 0;  List\* link;  int count = 0;  int sum = 0;  public:  Stack(){}  Stack(int a) {  int digit;  cout << "Enter " << a << " numbers: ";  for (int i = 0; i < a; i++) {  cin >> digit;  push(digit);  }  }  void push(int value) {  bool lock=false;  for (List\* i=first; i!=0; i=i->next){  if (value==i->data){  lock = true;  break;  }  }  if (!lock){  count++;  sum += value;  link = new List;  link->counter = count;  link->data = value;  link->next = 0;  if (last == 0) {  link->prev = 0;  first = last = link;  }  else  {  last->next = link;  link->prev = last;  }  last = link;  }  }  void pop() {  sum -= last->data;  last = last->prev;  delete last->next;  count--;  }  int total() { return count; }  double average() { return (double)sum / count; }  int min(int a = 0) {  int min=last->data;  int value=last->counter;  for (List\* i = last->prev; i != 0; i = i->prev) {  if (min > i->data) {  min = i->data;  value = i->counter;  }  }  if (a) return value;  return min;  }  int max() {  int max = last->data;  for (List\* i = last->prev; i != 0; i = i->prev) {  if (max < i->data) max = i->data;  }  return max;  }  int number(int value){  if (value > count) throw "Out Of Bounds";  List\* j = first;  for (int i = 1; i < value; i++){  j = j->next;  }  return j->data;  }  ~Stack() {  while (last) {  link = last;  last = last->prev;  delete link;  }  }  };  int main(){  Stack a(4);  cout << endl;  cout << "Total: " << a.total() << endl;  cout << "Average: " << a.average() << endl;  cout << "Min: " << a.min() << endl;  cout << "Max: " << a.max() << endl;  try{  cout << "Number #4: " << a.number(4) << endl;  }  catch(const char\* ex ){  cout << ex << endl;  }  cout << "The last before min: " << a.number(a.min(1)-1) << endl;  cout << endl;  system("pause");  return 0;  } |

Приклад виконання програми:

|  |
| --- |
| Enter 4 numbers: 1 4 8 -9  Total: 4  Average: 1  Min: -9  Max: 8  Number #4: -9  The last before min: 8  Press any key to continue . . . |

Висновок

Я навчився застосовувати стек – динамічну множину для виконання необхідних мені завданнь. Стек – це динамічна множина, її основний принцип роботи це дотримання правила LIFO (Останнім зайшов – Першим вийшов), тобто ми можемо додавати ті видаляти елементи тільки з кінця (tail). У коді це реалізовано методами push та pop. Сам стек реалізован за допомогою двозв'язаного списка.