

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 1

з дисципліни “Основи програмування”

тема “Модулі та C++ класи”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-92  Мовчан Максим Олександрович  (*прізвище, ім’я, по батькові*)  варіант № 9 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Ознайомитись з особливостями мови С++ та класами.  
Навчитись реалізовувати основні колекції елементів різними способами.  
Навчитись створювати програмні модулі.  
Навчитись розбивати код проекту на декілька файлів і компілювати його.

**Постановка завдання**

Програма повинна:

1. Зчитати всі дробові числа з текстового файлу і додати їх у список.
2. Вивести вміст списку у консоль.
3. Перенести всі числа, модуль яких більший за 10, у кінець списку.
4. Вивести вміст списку у консоль.
5. Створити два екземпляри даного АТД.
6. Переписати всі елементи непарних позицій списку у перший стек, а всіх парних - у другий.
7. Вивести вміст обох екземплярів АТД у консоль.
8. Створити новий список і переписати у нього всі елементи з першого стека, а потім з другого.
9. Вивести вміст результуючого списку у консоль.

**Інтерфейси модулів**

|  |
| --- |
| **class** DynamicArray { //динамічний масив  **float** \*\_items;  size\_t \_capacity;  **public**:  DynamicArray(size\_t initialCapacity);  ~DynamicArray();  **int** get(**int** index); //повернути число за індексом  **void** set(**int** index, **float** value); //записати число у комірку за індексом  size\_t size(); //повернути кількість значущих елементів массиву  **void** resize(size\_t newCapacity); //змінити кількість виділеної пам’яті  }; |

|  |
| --- |
| **class** List { //список на основі динамічного масиву  DynamicArray \_array;  size\_t \_size;  **public**:  List();  ~List();  size\_t size(); //повернути кількість елементів списку  **float** get(**int** index); //повернути елемент списку за індексом  **void** set(**int** index, **float** value); //записати нове число за індексом  **void** insert(**int** index, **float** value); //вставити елемент на індекс, зсунути все вправо  **void** remove\_at(**int** index); //видалити елемент списку за індексом  **void** push\_back(**float** value); //додати новий елемент у кінець списку  **void** remove(**float** value); //видалити за значенням  **int** index\_of(**float** value); //повернути індекс елемента за значенням  **bool** contains(**float** value); //перевірити, чи містить список елемент  **bool** empty(); //перевірка чи є в списку числа  **void** clear(); //очистити список  **void** print(); //вивести вміст списку у консоль  }; |
| **class** Stack { //стек на основі динамічного масиву  DynamicArray \_array;  size\_t \_size;  **public**:  Stack();  ~Stack();  size\_t size(); //повернути кількість елементів стеку  **void** push (**float** value); //додати новий елемент на верх (кінець) стеку  **float** peek(); //повернути верхній (останній) елемент стеку  **float** pop(); //видалити верхній (останній) елемент стеку  **bool** empty(); //перевірка чи є в стеку числа  **void** print(); //вивести вміст стеку у консоль  }; |

|  |  |
| --- | --- |
| data.txt | console output |
| 1  -333.333  1.1  -22.22  -1  333.333  -1.1  22.22 | [xon@dev build]$ ./lab1.out ../data.txt  LIST AT BEGIN  \_\_\_\_\_\_\_\_\_\_\_\_\_  List  0) 1  1) -333.333  2) 1.1  3) -22.22  4) -1  5) 333.333  6) -1.1  7) 22.22  SIZE 8  CAPACITY: 16  LIST AFTER PUSHING BACK  \_\_\_\_\_\_\_\_\_\_\_\_\_  List  0) 1  1) 1.1  2) -1  3) -1.1  4) -333.333  5) -22.22  6) 333.333  7) 22.22  SIZE 8  CAPACITY: 16  STACK1 (odd indexes)  \_\_\_\_\_\_\_\_\_\_\_\_\_  Stack:  0) 22.22  1) -22.22  2) -1.1  3) 1.1  STACK2 (even indexes)  \_\_\_\_\_\_\_\_\_\_\_\_\_  Stack:  0) 333.333  1) -333.333  2) -1  3) 1  LIST AFTER MERGE  \_\_\_\_\_\_\_\_\_\_\_\_\_  List  0) 22.22  1) -22.22  2) -1.1  3) 1.1  4) 333.333  5) -333.333  6) -1  7) 1  SIZE 8  CAPACITY: 16  END  \_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Тексти коду програм**

|  |
| --- |
| **main.cpp** |
| #include "MODULES/fs.h"  #include "MODULES/list.h"  #include "MODULES/stack.h"  #include <cstdlib>  #include <iostream>  using namespace std;  void connect(Stack \*stack\_first, Stack \*stack\_second, List \*sh\_list)  {  int tmp = stack\_first->size();  for (int i = 0; i < tmp; i++)  {  sh\_list->push\_back(stack\_first->pop());  }  tmp = stack\_second->size();  for (int i = 0; i < tmp; i++)  {  sh\_list->push\_back(stack\_second->pop());  }  }  void push\_back(List \*abl)  {  int j = 0;  for (int i = 0; i < abl->size(); i++)  {  if (abs(abl->get(j)) > 10)  {  float tmp = abl->get(j);  abl->remove\_at(j);  abl->push\_back(tmp);  }  else  {  j++;  }  }  }  void to\_stack(List \*abl, Stack \*stack, int delta)  {  for (int i = delta; i < abl->size(); i += 2)  {  stack->push(abl->get(i));  }  }  int main(int argc, char \*argv[])  {  check\_args(argc, 2);  FILE \*fin = fopen(argv[1], "rw");  check\_open(fin);  List list;  float tmp = 0;  while (fscanf(fin, "%f", &tmp) >= 0)  {  list.push\_back(tmp);  }  cout << "\033[;;47mLIST AT BEGIN\033[0m" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  list.print();  push\_back(&list);  cout << "\033[;;47mLIST AFTER PUSHING BACK\033[0m" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  list.print();  Stack stack1;  Stack stack2;  to\_stack(&list, &stack1, 1);  to\_stack(&list, &stack2, 0);  cout << "\033[;;47mSTACK1 (odd indexes)\033[0m" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  stack1.print();  cout << "\033[;;47mSTACK2 (even indexes)\033[0m" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  stack2.print();  List mergedlist;  connect(&stack1, &stack2, &mergedlist);  cout << "\033[;;47mLIST AFTER MERGE\033[0m" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  mergedlist.print();  cout << "\033[;;47mEND\033[0m" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  check\_close(fin);  cout << endl;  } |
| **dynarray.cpp** |
| #include "dynamic\_array.h"  DynamicArray::DynamicArray( int capacity ) {  items\_ = new float[capacity];  capacity\_ = capacity;  }  DynamicArray::~DynamicArray() {  delete[] items\_;  }  float DynamicArray::get( int index ) {  return items\_[index];  }  void DynamicArray::set( int index, float value ) {  items\_[index] = value;  }  void DynamicArray::resize( int capacity ) {  float \* tmp = new float[capacity];  for( int i = 0; i < capacity\_; i++ ) tmp[i] = items\_[i];  delete items\_;  items\_ = tmp;  capacity\_ = capacity;  }  int DynamicArray::size() {  return capacity\_;  } |
| **list.cpp** |
| #include "list.h"  List::List(): \_array{ 4 } {}  List::~List() {}  int List::size() {  return size\_;  }  float List::get( int index ) {  if( size\_ >= index && index >= 0 ) {  return \_array.get( index );  }  perror( "\033[;;41mWRONG INDEX\033[0m\n" );  exit( EXIT\_FAILURE );  }  void List::set( int index, float value ) {  if( size\_ >= index && index >= 0 ) {  \_array.set( index, value );  return;  }  perror( "\033[;;41mWRONG INDEX\033[0m\n" );  exit( EXIT\_FAILURE );  }  void List::insert( int index, float value ) {  if( size\_ == \_array.size() - 1 ) \_array.resize( \_array.size() \* 2 );  for( int i = 0; index + i < size\_; i++ ) {  set( size\_, get( index + i ) );  }  set( index, value );  }  void List::remove\_at( int index ) {  for( int i = index; i < size\_ - 1; i++ ) {  set( i, get( i + 1 ) );  }  size\_--;  }  void List::push\_back( float value ) {  if( size\_ == \_array.size() - 1 ) \_array.resize( \_array.size() \* 2 );  set( size\_, value );  size\_++;  }  void List::remove( float value ) {  for( int i = 0; i < size\_; i++ ) {  if( get( i ) == value ) {  remove\_at( i );  }  }  }  int List::index\_of( float value ) {  for( int i = 0; i < size\_; i++ ) {  if( get( i ) == value ) return i;  }  return -1;  }  bool List::contains( float value ) {  for( int i = 0; i < size\_; i++ ) {  if( get( i ) == value ) return true;  }  return false;  }  bool List::empty() {  return size\_ == 0;  }  void List::clear() {  size\_ = 0;  }  void List::print() {  cout << "\033[;;44mList\033[0m";  for( int i = 0; i < size\_; i++ ) {  cout << "\n\033[;;46m" << i << ")\033[0m " << get( i );  }  cout << "\033[0m" << endl;  cout << "\033[;;44mSIZE\033[0m " << size();  cout << "\033[0m" << endl;  cout << "\033[;;44mCAPACITY:\033[0m " << \_array.size() << endl;  cout << endl;  } |
| **stack.cpp** |
| #include "stack.h"  Stack::Stack(): \_array{ 4 } {}  Stack::~Stack() {}  int Stack::size() {  return size\_;  }  void Stack::push( float value ) {  \_array.set( size\_, value );  size\_++;  if( size\_ == \_array.size() ) \_array.resize( \_array.size() \* 2 );  }  float Stack::peek() {  if( empty() ) {  cout << "\033[;;41mSTACK IS EMPTY\033[0m\n" << endl;  exit( EXIT\_FAILURE );  } else  return \_array.get( size\_ - 1 );  }  float Stack::pop() {  if( empty() ) {  cout << "\033[;;41mSTACK IS EMPTY\033[0m\n" << endl;  exit( EXIT\_FAILURE );  }  float tmpVal = \_array.get( size\_ - 1 );  size\_--;  return tmpVal;  }  bool Stack::empty() {  return size\_ == 0;  }  void Stack::print() {  cout << "\033[;;42mStack:\033[0m ";  for( int i = 0; i < size\_; i++ ) {  cout << "\n\033[;;46m" << i << ")\033[0m " << \_array.get( size\_ - 1 - i );  }  cout << endl << endl;  } |
| **fs.cpp** |
| #include "fs.h"  void check\_args( int arg, int num ) {  if( arg != num ) {  perror( "\033[;;41mWRONG COUNT OF ARGUMENTS. EXITING\033[0m\n" );  exit( EXIT\_FAILURE );  }  }  void check\_open( FILE \* file ) {  if( !file ) {  perror( "\033[;;41mCAN'T OPEN FILE. EXITING\033[0m\n" );  exit( EXIT\_FAILURE );  }  }  void check\_close( FILE \* file ) {  if( fclose( file ) == EOF ) {  perror( "\033[;;41mCAN'T CLOSE FILE. EXITING\033[0m\n" );  exit( EXIT\_FAILURE );  }  } |

**Висновки**

У лабораторній роботі були використані структури даних на основі динамічного масиву. Виділення пам’яті було виконано за допомогою оператора new та звільнена за допомогою оператор delete[].

Для виводу у консоль був використаний заголовочний файл <iostream>, об’єкти cout, endl з простору імен std та оператор вставки у потік <<.

Компіляція коду відбувалася за допомогою компілятора g++.