**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. |  | Полуянов В. Н. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение заголовочных файлов. Написание собственных заголовочных файлов и их последующие включение в исходный код. Включение заголовочных файлов из других каталогов.

**Основные теоретические положения.**

Заголовочный файл – файл, содержимое которого автоматически добавляется препроцессором в исходный текст в том месте, где располагается некоторая директива. Это основной способ подключить к программе типы данных, структуры, функции, используемые в другом модуле и др.

По умолчанию используется расширение .h, иногда для заголовочных файлов языка программирования C++ используют расширение .hpp.

В работе с заголовочным файлом следует применять директивы #ifndef, #define, #endif.

Директива #ifndef означает «если не определено». #endif, соответственно, определяет конец блока условия.

Директива #define определяет идентификатор и последовательность символов, которой будет замещаться данный идентификатор при его обнаружении в тексте программы.

Таким образом, как только подключается заголовок, препроцессор проверяет, определено ли уникальное значение и, если он не определен, определяет его и переходит к остальной части страницы. Также это предотвращает повторное объявление любых идентификаторов, таких как типы данных, перечисления и статические переменные.

**Постановка задачи.**

Создать модули на основе предыдущих практических работ и написать программу, которая:

1. Выводит интерактивное меню модулей (практических работ)
2. Ожидает выбор пользователя и запускает соответствующую его выбору практическую работу
3. По завершении работы с модулем возвращает на экран меню

**Выполнение работы.**

Код программы представлен в приложении А.

1. При запуске программы пользователю выводится меню доступных команд и ожидается ввод числа. (рис. 1).

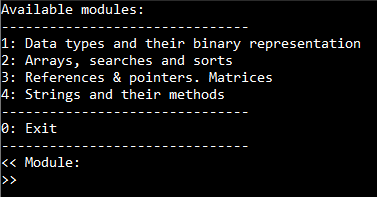


Рисунок . Запуск программы и главное меню

1. Следующий шаг зависит от введенной команды, если пользователь ввел:
   1. “0”, то выполнение программы завершается.
   2. “1”, то запускается практическая работа №1 «Типы данных и их внутреннее представление в памяти» (рис. 2)

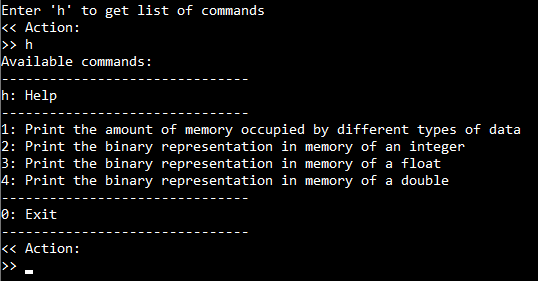


Рисунок 2. Модуль 1

* 1. “2”, то запускается практическая работа №2 «Одномерные статические массивы» (рис. 3).

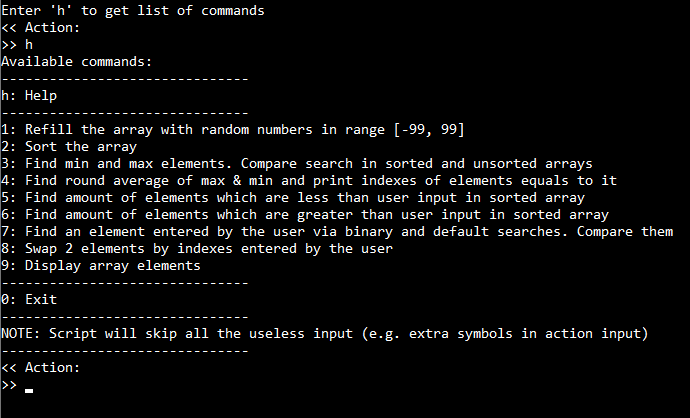


Рисунок 3. Модуль 2

* 1. “3”, то запускается практическая работа №3 «Двумерные статические массивы. Указатели» (рис. 4).

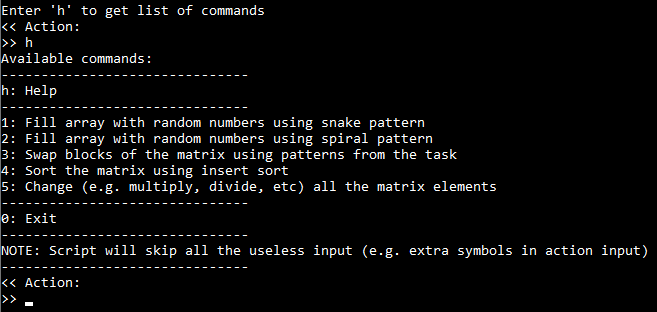


Рисунок 4. Модуль 3

* 1. “4”, то запускается практическая работа №4 «Текстовые строки» (рис. 5).

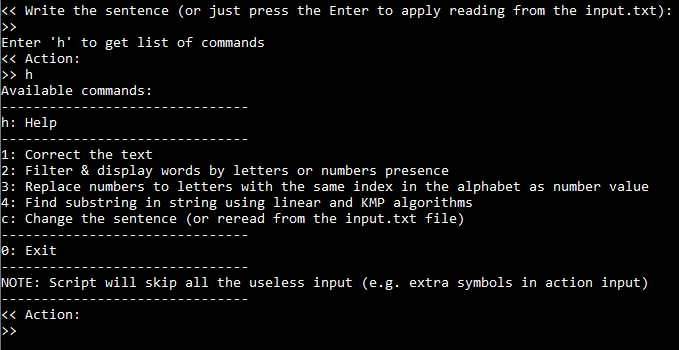


Рисунок 5. Модуль 4

* 1. Любой символ или набор символов, отличный от существующий команд, то будет выведено сообщение об ошибке (рис. 6).



Рисунок 8. Неизвестная команда

**Выводы.**

В ходе работы были изучены заголовочные файлы и способы их создания, методы создания собственных модулей и их последующего импортирования в исходный код.

Приложение А

рабочий код

#include <iostream>  
#include <iomanip>  
#include <windows.h>  
#include "modules/p1-datatypes.h"  
#include "modules/p2-arrays.h"  
#include "modules/p3-pointers.h"  
#include "modules/p4-strings.h"  
  
int main() {  
 while (true) {  
 *// Help* system("cls");  
 std::cout << "Available modules:\n";  
 std::cout << std::setw(32) << std::setfill('-') << '\n';  
 std::cout << "1: Data types and their binary representation\n";  
 std::cout << "2: Arrays, searches and sorts\n";  
 std::cout << "3: References & pointers. Matrices\n";  
 std::cout << "4: Strings and their methods\n";  
 std::cout << std::setw(32) << std::setfill('-') << '\n';  
 std::cout << "0: Exit\n";  
 std::cout << std::setw(32) << std::setfill('-') << '\n';  
 std::cout << std::setfill(' ');  
  
 *// Get command from user* char userAction;  
 std::cout << "<< Module:\n>> ";  
 std::cin.sync();  
 std::cin >> userAction;  
  
 *// Exit* if (userAction == '0') break;  
  
 system("cls");  
 switch (userAction) {  
 case '1': {  
 dataTypes();  
 break;  
 }  
  
 case '2': {  
 arrays();  
 break;  
 }  
  
 case '3': {  
 pointers();  
 break;  
 }  
  
 case '4': {  
 strings();  
 break;  
 }  
  
 *// Unknown module error* default: {  
 std::cout << "RuntimeError: unknown module\n";  
 Sleep(500);  
 }  
 }  
 }  
  
 return 0;  
}