Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

**Отчет по лабораторной работе №6**

**По дисциплине**

«Основы программной инженерии»

На тему:

«Модульное программирование»

**Выполнил:**

Рутковский Никита

Викторович, ФИТ

ПИ-6, 1 курс

**руководитель:**

Наркевич Аделина

Сергеевна

Минск 2024

**Модульное программирование**

**Цель работы:** ознакомление с основными парадигмами программирования. Введение Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями. Модуль – функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом. Функциональная декомпозиция задачи – разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач – модулей.

**1.** Используйте при выполнении лабораторной работы материал лекции 7.

**2.** Отчет по лабораторной работе оформить в виде документа в MS Word.

**3. Задание.** Для задачи из п.5 лабораторной работы 5:  
**1)** Дополнительно предусмотреть возможность ввода с клавиатуры нескольких символов последовательно.

#include <iostream>

#include <cctype>

#include <windows.h>

#include <string>

**void** showMenu() {

std::cout << "Выберите вариант использования:\n";

std::cout << "1 - Разница кодов в ASCII для латинских букв\n";

std::cout << "2 - Разница кодов в Windows-1251 для русских букв\n";

std::cout << "3 - Код символа для введенной цифры\n";

std::cout << "4 - Выход\n";

}

std::string getInputString() {

std::string input;

std::cout << "Введите строку символов: ";

std::cin >> input;

**return** input;

}

**int** getOption() {

**int** option;

std::cout << "Введите вариант использования: ";

std::cin >> option;

**return** option;

}

**void** processOption(**int** option, **const** std::string& input) {

**switch** (option) {

**case** 1:

**for** (**char** ch : input) {

**if** (isalpha(ch) && isascii(ch)) {

**char** upper = toupper(ch);

**char** lower = tolower(ch);

std::cout << "Разница кодов в ASCII для '" << ch << "': " << abs(**int**(upper) - **int**(lower)) << "\n";

}

**else** {

std::cout << "Ошибка: введен некорректный символ '" << ch << "' для варианта 1\n";

}

}

**break**;

2) Выполнить постановку задачи.   
напишите диалоговую программу, которая в зависимости от выбранного варианта использования выполняет действия пунктов 1, 2, 3 для любого введенного с клавиатуры символа. Варианты использования: 1 – определение разницы значений кодов в ASCII буквы в прописном и строчном написании, если введен символ латинского алфавита, иначе вывод сообщения об ошибке; 2 – определение разницы значений кодов в Windows-1251 буквы в прописном и строчном написании, если введен символ русского алфавита, иначе вывод сообщения об ошибке; 3 – вывод в консоль кода символа, соответствующего введенной цифре, иначе вывод сообщения об ошибке; 4 – выход из программы. Текст сообщения об ошибке произвольный.   
3) Определить входные и выходные данные:  
  
Модуль showMenu:  
Входных данных нет, выходные данные – текстовое меню.  
  
Модуль getInputString:  
Входные данные – строка, введенная пользователем, выходные данные – возврат строки для дальнейшей работы с ней.  
  
Модуль getOption:  
Входные данные – число, выходные – возврат числа для выбора варианта использования.  
  
Модуль processOption:  
Входные данные – строка из модуля getinputstring, число их модуля getoption, выходные – возврат результата в зависимости от приведенных входных данных.  
  
Модуль main  
Входные данные – нет, выходные данные – нет.

<< "' для варианта 1\n";

}

}

**break**;

**case** 2:

**for** (**char** ch : input) {

**if** ((ch >= '\xC0' && ch <= '\xDF') || (ch >= '\xE0' && ch <= '\xFF')) {

std::cout << "Код символа '" << ch << "': " << std::hex << std::uppercase << **static\_cast**<**int**>(ch) << std::endl;

**char** convertedSymb;

**if** (ch >= '\xC0' && ch <= '\xDF') {

// Прописная буква

convertedSymb = ch + 0x20; // Преобразование в строчную букву

}

**else** {

// Строчная буква

convertedSymb = ch - 0x20; // Преобразование в прописную букву

}

std::cout << "Преобразованный символ: " << convertedSymb << std::endl;

std::cout << "Код преобразованного символа: " << std::hex << std::uppercase << **static\_cast**<**int**>(convertedSymb) << std::endl;

std::cout << "Разница между кодами: " << std::dec << abs(**static\_cast**<**int**>(ch) - **static\_cast**<**int**>(convertedSymb)) << std::endl;

}

**else** {

std::cout << "Символ '" << ch << "' не является кириллическим." << std::endl;

}

}

**break**;

**case** 3:

**for** (**char** ch : input) {

**if** (isdigit(ch)) {

std::cout << "Код символа '" << ch << "': " << **int**(ch) << "\n";

}

**else** {

std::cout << "Ошибка: введен некорректный символ '" << ch << "' для варианта 3\n";

}

}

**break**;

**case** 4:

std::cout << "Выход из программы.\n";

exit(0);

**default**:

std::cout << "Ошибка: некорректный вариант использования\n";

}

}

**int** main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

**while** (**true**) {

showMenu();

**int** option = getOption();

**if** (option == 4) {

processOption(option, "");

}

**else** {

std::string input = getInputString();

processOption(option, input);

}

}

**return** 0;

}

std::string input = getInputString();

processOption(option, input);

}

}

**return** 0;

}

4) Блок-схема:

Изображение выглядит как текст, чек, снимок экрана, Шрифт

Автоматически созданное описание

5) **Модули программы:**

1. **Модуль отображения меню (**showMenu**)**
   * **Назначение:** Отображение меню с вариантами использования.
   * **Входные данные:** Нет.
   * **Выходные данные:** Текстовое меню, отображаемое в консоли.
2. **Модуль получения ввода пользователя (**getInputString**и**getInputChar**)**
   * **Назначение:** Получение строки символов или одного символа от пользователя.
   * **Входные данные:** Нет.
   * **Выходные данные:** Введенная пользователем строка символов или символ.
3. **Модуль получения варианта использования (**getOption**)**
   * **Назначение:** Получение выбранного пользователем варианта использования.
   * **Входные данные:** Нет.
   * **Выходные данные:** Введенный пользователем вариант использования (целое число).
4. **Модуль обработки вариантов использования (**processOption**)**
   * **Назначение:** Выполнение действий в зависимости от выбранного варианта использования и введенной строки.
   * **Входные данные:** Вариант использования (целое число) и введенная строка символов или символ.
   * **Выходные данные:** Результаты выполнения выбранного варианта использования, выводимые в консоль.
5. **Основной модуль (**main**)**
   * **Назначение:** Основной цикл программы, который управляет выполнением всех остальных модулей.
   * **Входные данные:** Нет.
   * **Выходные данные:** Нет.

6) **Модуль**showMenu

**Состав:**

* Функция showMenu.

**Назначение:**

* Выводит на экран меню с вариантами использования программы.

**Входные данные:**

* Нет.

**Выходные данные:**

* Текстовое меню, отображаемое в консоли.

**Алгоритм:**

1. Вывести текст с вариантами использования программы.

**Модуль**getInputString

**Состав:**

* Функция getInputString.

**Назначение:**

* Запрашивает у пользователя ввод строки символов.

**Входные данные:**

* Нет.

**Выходные данные:**

* Введенная пользователем строка символов.

**Алгоритм:**

1. Вывести приглашение для ввода строки.
2. Прочитать строку, введенную пользователем.
3. Вернуть введенную строку.

**Модуль**getOption

**Состав:**

* Функция getOption.

**Назначение:**

* Запрашивает у пользователя выбор варианта использования.

**Входные данные:**

* Нет.

**Выходные данные:**

* Введенный пользователем вариант использования (целое число).

**Алгоритм:**

1. Вывести приглашение для ввода варианта использования.
2. Прочитать введенное пользователем число.
3. Вернуть введенное число.

**Модуль**processOption

**Состав:**

* Функция processOption.

**Назначение:**

* Выполняет действия в зависимости от выбранного варианта использования и введенной строки.

**Входные данные:**

* Вариант использования (целое число).
* Введенная строка символов (строка).

**Выходные данные:**

* Результаты выполнения выбранного варианта использования, выводимые в консоль.

**Алгоритм:**

1. В зависимости от значения option:
   * **Если**option**== 1:**
     1. Для каждого символа в строке:
        + Если символ является латинской буквой:
          - Вычислить разницу кодов в ASCII между прописной и строчной формами.
          - Вывести результат.
        + Иначе вывести сообщение об ошибке.
   * **Если**option**== 2:**
     1. Для каждого символа в строке:
        + Если символ является русской буквой в кодировке Windows-1251:
          - Преобразовать символ в противоположный регистр.
          - Вычислить разницу кодов между исходным и преобразованным символами.
          - Вывести результат.
        + Иначе вывести сообщение об ошибке.
   * **Если**option**== 3:**
     1. Для каждого символа в строке:
        + Если символ является цифрой:
          - Вывести код символа.
        + Иначе вывести сообщение об ошибке.
   * **Если**option**== 4:**
     1. Вывести сообщение о выходе из программы.
     2. Завершить выполнение программы.
   * **Иначе:**
     1. Вывести сообщение об ошибке.

**Модуль**main

**Состав:**

* Функция main.

**Назначение:**

* Основной цикл программы, который управляет выполнением всех остальных модулей.

**Входные данные:**

* Нет.

**Выходные данные:**

* Нет.

**Алгоритм:**

* 1. Вызвать функцию showMenu для отображения меню.
  2. Вызвать функцию getOption для получения варианта использования.
  3. Если вариант использования равен 4, вызвать processOption с пустой строкой и завершить программу.
  4. Иначе, вызвать функцию getInputString для получения строки символов.
  5. Вызвать функцию processOption с полученными данными.

7) Нисходящее проектирование

1. **Определение целей:**
   * Программа должна предоставлять пользователю меню с вариантами использования и выполнять соответствующие действия в зависимости от выбранного варианта.
2. **Разработка общей структуры:**
   * Основные компоненты: меню, ввод данных, обработка вариантов использования, вывод результатов.
3. **Декомпозиция на подзадачи:**
   * Меню: функция для отображения меню.
   * Ввод данных: функции для получения варианта использования и ввода символов.
   * Обработка вариантов использования: функции для обработки каждого варианта (ASCII, Windows-1251, цифры, выход).
   * Вывод результатов: функции для вывода результатов или сообщений об ошибках.
4. **Детализация и реализация:**
   * Реализуйте функции showMenu, getOption, getInputString, processOption.
   * Реализуйте основной цикл программы в функции main, который будет вызывать остальные функции в зависимости от выбранного варианта.
5. **Тестирование и отладка:**
   * Тестируйте каждую функцию отдельно, чтобы убедиться в их корректной работе.
   * Тестируйте программу в целом, чтобы убедиться, что все компоненты взаимодействуют правильно

8)

Изображение выглядит как текст, снимок экрана, чек, число

Автоматически созданное описание

**Дополнительное задание:  
1)** Сделаем простую программу для сложения двух чисел с помощью модульного программирования

2. Для задания по своему варианту 4-ой лабораторной работы:  
Постановка задачи: Написать программу, которая будет выводить название дня недели по введенному номеру дня.  
**Модули программы:**

#include <iostream>

// Модуль ввода чисел

**void** inputNumbers(**int**& a, **int**& b) {

std::cout << "Введите первое число: ";

std::cin >> a;

std::cout << "Введите второе число: ";

std::cin >> b;

}

// Модуль сложения чисел

**int** addNumbers(**int** a, **int** b) {

**return** a + b;

}

// Модуль вывода результата

**void** outputResult(**int** result) {

std::cout << "Результат сложения: " << result << std::endl;

}

// Основной модуль

**int** main() {

setlocale(LC\_ALL, "Russian");

**int** num1, num2;

inputNumbers(num1, num2);

**int** sum = addNumbers(num1, num2);

outputResult(sum);

**return** 0;

}

1. **Модуль ввода номера дня недели (**inputDayNumber**)**
2. **Модуль получения названия дня недели (**getDayName**)**
3. **Модуль вывода названия дня недели (**outputDayName**)**
4. **Основной модуль (**main**)**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Модуль**inputDayNumber

**Состав:**

* Функция inputDayNumber.

**Назначение:**

* Запрашивает у пользователя ввод номера дня недели.

**Входные данные:**

* Нет.

**Выходные данные:**

* Введенный пользователем номер дня недели (целое число).

**Алгоритм:**

1. Вывести сообщение “Введите номер дня недели (1-7):”.
2. Прочитать введенное пользователем число.
3. Вернуть введенное число.

**Модуль**getDayName

**Состав:**

* Функция getDayName.

**Назначение:**

* Возвращает название дня недели по его номеру.

**Входные данные:**

* Номер дня недели (целое число).

**Выходные данные:**

* Название дня недели (строка).

**Алгоритм:**

1. В зависимости от значения dayNumber:
   * Если dayNumber == 1, вернуть “Понедельник”.
   * Если dayNumber == 2, вернуть “Вторник”.
   * Если dayNumber == 3, вернуть “Среда”.
   * Если dayNumber == 4, вернуть “Четверг”.
   * Если dayNumber == 5, вернуть “Пятница”.
   * Если dayNumber == 6, вернуть “Суббота”.
   * Если dayNumber == 7, вернуть “Воскресенье”.
   * Иначе вернуть “Некорректный номер дня недели”.

**Модуль**outputDayName

**Состав:**

* Функция outputDayName.

**Назначение:**

* Выводит название дня недели на экран.

**Входные данные:**

* Название дня недели (строка).

**Выходные данные:**

* Нет.

**Алгоритм:**

1. Вывести сообщение "Название дня недели: " и значение dayName.

**Основной модуль**main

**Состав:**

* Функция main.

**Назначение:**

* Управляет выполнением всех остальных модулей.

**Входные данные:**

* Нет.

**Выходные данные:**

* Нет.

**Алгоритм:**

1. Вызвать функцию inputDayNumber для получения номера дня недели.
2. Вызвать функцию getDayName с полученным номером дня недели.
3. Вызвать функцию outputDayName с полученным названием дня недели.

**Разработка общей структуры:**

Основные компоненты:

1. Ввод данных
2. Обработка данных
3. Вывод результатов

**Декомпозиция на подзадачи:**

1. **Ввод данных**: функция для получения номера дня недели.
2. **Обработка данных**: функция для получения названия дня недели по его номеру.
3. **Вывод результатов**: функция для вывода названия дня недели.

**Детализация и реализация:**

1. Реализуйте функцию inputDayNumber, которая запрашивает у пользователя номер дня недели.
2. Реализуйте функцию getDayName, которая возвращает название дня недели по его номеру.
3. Реализуйте функцию outputDayName, которая выводит название дня недели.
4. Реализуйте основной цикл программы в функции main, который будет вызывать остальные функции в зависимости от введенного номера дня недели.

**Тестирование и отладка:**

1. Тестируйте каждую функцию отдельно, чтобы убедиться в их корректной работе.
2. Тестируйте программу в целом, чтобы убедиться, что все компоненты взаимодействуют правильно.

**Код программы:**

#include <iostream>

#include <string>

// Модуль ввода номера дня недели

**int** inputDayNumber() {

**int** dayNumber;

std::cout << "Введите номер дня недели (1-7): ";

std::cin >> dayNumber;

**return** dayNumber;

}

// Модуль получения названия дня недели

std::string getDayName(**int** dayNumber) {

**switch** (dayNumber) {

**case** 1: **return** "Понедельник";

**case** 2: **return** "Вторник";

**case** 3: **return** "Среда";

**case** 4: **return** "Четверг";

**case** 5: **return** "Пятница";

**case** 6: **return** "Суббота";

**case** 7: **return** "Воскресенье";

**default**: **return** "Некорректный номер дня недели";

}

}

// Модуль вывода названия дня недели

**void** outputDayName(**const** std::string& dayName) {

std::cout << "Название дня недели: " << dayName << std::endl;

}

// Основной модуль

**int** main() {

setlocale(LC\_ALL, "Russian");

**int** dayNumber = inputDayNumber();

std::string dayName = getDayName(dayNumber);

outputDayName(dayName);

**return** 0;

}

1. **Вопросы:**Определение парадигмы программирования и основные парадигмы:

setlocale(LC\_ALL, "Russian");

**int** dayNumber = inputDayNumber();

std::string dayName = getDayName(dayNumber);

outputDayName(dayName);

**return** 0;

}

* + Парадигма программирования — это совокупность понятий, идей и правил, которые определяют подход к написанию кода. Основные парадигмы включают:
    - Императивная (например, процедурное и объектно-ориентированное программирование)
    - Декларативная (например, функциональное и логическое программирование)
    - Логическая
    - Процедурная

1. Методология разработки программного обеспечения:
   * Методология разработки ПО — это совокупность методов, применяемых на различных стадиях жизненного цикла программного обеспечения и имеющих общий философский подход. Назначение методологий — структурировать процесс разработки, улучшить управление проектом и повысить качество конечного продукта.
2. Отличие императивного программирования от декларативного:
   * Императивное программирование описывает, как должна выполняться задача, с указанием конкретных шагов и последовательности действий.
   * Декларативное программирование описывает, что должно быть достигнуто, без указания конкретных шагов выполнения.
3. Цели и основные принципы структурного программирования:
   * Цели: улучшение читаемости, модульности и сопровождения кода.
   * Принципы: абстракция, модульность и последовательное выполнение.
4. Базовые конструкции языка программирования в структурном программировании:
   * Последовательность: выполнение действий по порядку.
   * Ветвление: выбор действий в зависимости от условий.
   * Цикл: повторение действий до достижения определенного условия.
5. Методы структурного программирования:
   * Декомпозиция: разделение задачи на подзадачи.
   * Модульность: создание независимых модулей.
   * Использование подпрограмм: функции и процедуры для повторного использования кода.
6. Принцип модульного проектирования:
   * Основной принцип — разделение системы на независимые модули, которые могут быть разработаны и протестированы отдельно.
7. Функциональная декомпозиция задачи:
   * Это процесс разделения сложной задачи на более простые подзадачи, каждая из которых решается отдельно.
8. Основные характеристики модуля:
   * Самодостаточность: модуль выполняет определенную функцию.
   * Интерфейс: четко определенные входы и выходы.
   * Повторное использование: модуль может быть использован в разных частях программы.
9. Методы модульного программирования:
   * Инкапсуляция: скрытие внутренней реализации модуля.
   * Абстракция: выделение важных характеристик и игнорирование несущественных деталей.
   * Композиция: объединение модулей для создания более сложных систем