Первая практика:

Задание 4: Структуры и динамическая память

Описание: Создайте структуру для представления точки в 2D-пространстве и выделите память для массива из 5 таких точек.

Требования:

Определите структуру Point с полями x и y.

Используйте malloc для выделения памяти под массив из 5 точек.

Считайте координаты точек с помощью ввода и выведите их на экран.

Задание 5: Статическая и динамическая память

Описание: Напишите программу, которая сравнивает использование статической и динамической памяти.

Требования:

Создайте массив с фиксированным размером (int staticArray[10]).

Сравните его с динамически выделенным массивом (int \*dynamicArray = malloc(10 \* sizeof(int))).

Выведите размеры обоих массивов с помощью sizeof.

Задание 6: Организация структуры проекта

Требования:

Создайте проект, который будет состоять из следующих файлов:

main.c

string\_utils.c

string\_utils.h

file\_utils.c

file\_utils.h

Задача:

В string\_utils.c реализуйте функции для работы со строками (например, подсчет длины строки, конкатенация).

В file\_utils.c реализуйте функции для работы с файлами (например, чтение из файла, запись в файл).

В main.c создайте меню, позволяющее пользователю выбирать, какую операцию выполнить (строковые или файловые операции).

Задание 7: Управление потоком программы

Описание: Реализуйте программу, которая проверяет, четное число или нечетное.

Требования:

Используйте условные операторы (if, else).

Запрашивайте у пользователя ввод числа.

Задание 8: Циклы и массивы

Описание: Напишите программу, которая запрашивает 5 чисел у пользователя и выводит их сумму.

Требования:

Используйте массив для хранения чисел.

Используйте цикл для суммирования значений.

Задание 9: Функции

Описание: Создайте программу, которая использует функции для выполнения математических операций (сложение, вычитание).

Требования:

Реализуйте две функции для операций.

Вызовите функции из main().

Задание 10: Завершение программы

Описание: Реализуйте программу, которая позволяет пользователю вводить числа до тех пор, пока он не введет 0, после чего программа должна выводить количество введенных чисел.

Требования:

Используйте цикл while.

Обеспечьте корректное завершение программы.

Вторая практика:

Задание 1: Массив указателей на функции (1)

Описание: Создайте массив указателей на функции, каждая из которых выполняет разные арифметические операции (сложение, вычитание, умножение и деление) над двумя числами.

Требования:

Определите функции: int add(int a, int b), int subtract(int a, int b), int multiply(int a, int b), float divide(int a, int b).

Создайте массив указателей на эти функции.

Напишите код, который запрашивает у пользователя два числа и выполняет каждую из операций с использованием указателей на функции.

Задание 2: Сравнение чисел с помощью указателей на функции (1)

Описание: Напишите программу, которая использует указатели на функции для сравнения двух чисел и возвращает большее из них.

Требования:

Определите функцию int max(int a, int b).

Создайте указатель на эту функцию.

Запросите у пользователя два числа и используйте указатель на функцию для нахождения и вывода большего числа.

Задание 3: Сортировка массива с использованием указателей на функции (2)

Описание: Реализуйте сортировку массива целых чисел, используя указатели на функции для определения порядка сортировки (по возрастанию или убыванию).

Требования:

Определите функции int compareAsc(int a, int b) и int compareDesc(int a, int b).

Напишите функцию сортировки, которая принимает массив и указатель на функцию сравнения.

Позвольте пользователю выбрать порядок сортировки и выведите отсортированный массив.

Задание 4: Применение указателей на функции в меню (3)

Описание: Создайте простое текстовое меню, которое позволяет пользователю выбрать одну из нескольких операций, реализованных с помощью указателей на функции.

Требования:

Реализуйте функции для различных операций (например, сложение, вычитание, умножение, деление).

Создайте указатель на функцию для каждой операции.

Отобразите меню с номерами операций и запросите выбор пользователя.

Выполните выбранную операцию, используя указатель на соответствующую функцию.

Третья практика:  
Задание 5: Система управления учебными курсами (1)

Цель: Создать многофайловый проект для управления учебными курсами.

Шаги:

• Создайте проект с файлами main.c, course.c, course.h и student.c, student.h.

• В course.h определите структуру Course, которая будет содержать информацию о курсе (название, код, количество студентов).

• В student.h определите структуру Student, которая будет содержать информацию о студенте (имя, фамилия, номер студенческого билета).

• Реализуйте функции для добавления студентов на курс и отображения информации о курсе. В main.c протестируйте вашу систему.

Задание 6: Простая игра "Угадай число" (1 - 2)

Цель: Реализовать игру с использованием многофайловой структуры.

Шаги:

• Создайте проект, состоящий из main.c, game.c и game.h.

• В game.h определите функции для инициализации игры, получения числа от пользователя и проверки угаданного числа.

• В game.c реализуйте логику игры, включая генерацию случайного числа и подсчет попыток.

• В main.c используйте функции из game.h для управления игрой.

Задание 7: Библиотека для работы с файлами (1)

Цель: Создать библиотеку для работы с текстовыми файлами.

Шаги:

• Создайте проект с файлами main.c, file\_utils.c и file\_utils.h.

• В file\_utils.h объявите функции для работы с файлами: чтение, запись и подсчет строк.

• В file\_utils.c реализуйте эти функции.

• В main.c используйте функции для чтения содержимого текстового файла и вывода его на экран.

Задание 8: Модульная калькуляторная программа (1)

Цель: Создать калькулятор с многофайловой структурой.

Шаги:

• Создайте проект с файлами main.c, calculator.c и calculator.h.

• В calculator.h объявите функции для основных арифметических операций (сложение, вычитание, умножение, деление).

• В calculator.c реализуйте эти функции и добавьте обработку ошибок (например, деление на ноль).

• В main.c создайте меню для выбора операции и взаимодействия с пользователем.

Задание 9: База данных студентов (1 – 2)

Цель: Реализовать простую базу данных для хранения информации о студентах.

Шаги:

• Создайте проект с файлами main.c, database.c и database.h.

• В database.h объявите структуру Student и функции для добавления, удаления и отображения студентов.

• В database.c реализуйте эти функции и используйте динамическое выделение памяти для хранения информации о студентах.

• В main.c создайте интерфейс для взаимодействия с базой данных (добавление, удаление и отображение студентов).

Четвёртая практика:  
Задание 4 (1) - оформленный отчет в красивом виде из задания 3

Задание 5 (2) - Использование динамического связывания для обработки данных

Описание: Создайте программу, которая загружает динамическую библиотеку для обработки строк. Библиотека должна содержать функции для конкатенации строк и вычисления длины строки

Задание 6 (2) - Динамическое связывание с использованием структуры

Описание: Создайте структуру для представления геометрических фигур (например, круг и квадрат). Реализуйте функции для вычисления площади и периметра, которые будут динамически загружаться из DLL.

Задание 7 (1) - Загрузка и использование нескольких DLL

Описание: Напишите программу, которая загружает несколько DLL, каждая из которых реализует разные арифметические операции, и позвольте пользователю выбрать, какую операцию использовать.

Пятая практика:  
Задание 4: Суммирование чисел в нескольких потоках (1)

Описание: Напишите программу, которая делит массив чисел на несколько частей и суммирует каждую часть в отдельном потоке. В конце программа должна выводить общую сумму.

Задание 5: Программа-производитель и программа-потребитель (2)

Описание: Реализуйте классическую задачу "производитель-потребитель" с использованием мьютексов и условных переменных. Создайте несколько потоков-производителей и потребителей.

Задание 6: Многопоточное копирование файлов (3)

Описание: Напишите программу, которая копирует содержимое файла в другой файл, используя несколько потоков. Каждый поток должен обрабатывать определенный сегмент исходного файла.

Задание 7: Счетчик простых чисел (1)

Описание: Напишите программу, которая находит простые числа в заданном диапазоне, используя несколько потоков. Каждый поток должен обрабатывать свою часть диапазона.

Задание 8: Скачивание веб-страниц (3)

Описание: Напишите программу, которая использует несколько потоков для скачивания веб-страниц. Каждый поток должен обрабатывать один URL и сохранять его содержимое в файл.

Шестая практика:

Задание 2: Создание классов и структур

Цель: Понять разницу между классами в C++ и структурами в C.

Описание:

Создайте структуру Person в C, которая будет содержать поля name и age.

Реализуйте функции для работы со структурой (инициализация, вывод информации).

Создайте класс Person в C++, который будет содержать те же поля и методы, что и структура в C.

Задание:

Сравните, как выглядит код в C и C++ для выполнения одинаковых задач, таких как создание объекта и вызов метода.

Задание 3: Наследование

Цель: Изучить концепцию наследования в C++ и ее отсутствие в C.

Описание:

В C++ создайте базовый класс Animal с методом speak().

Создайте производные классы Dog и Cat, которые будут переопределять метод speak().

Задание:

Реализуйте аналогичную функциональность в C, используя структуры и функции, но без наследования.

Сравните, как реализовать полиморфизм в C++ с использованием виртуальных функций и как можно имитировать это в C с помощью указателей на функции.

Задание 4: Инкапсуляция

Цель: Понять концепцию инкапсуляции и её реализацию в C++ и C.

Описание:

Создайте класс BankAccount в C++, который будет содержать защищенные поля balance и accountNumber, а также публичные методы для внесения и снятия средств.

Реализуйте аналогичную структуру в C с использованием глобальных переменных и функций для доступа к данным.

Задание:

Обсудите преимущества и недостатки инкапсуляции в C++ по сравнению с реализацией в C.

Задание 5: Шаблоны и обобщенное программирование

Цель: Изучить возможности обобщенного программирования в C++ и их отсутствие в C.

Описание:

Создайте шаблонный класс Container в C++, который будет хранить массив объектов и обеспечивать методы добавления, удаления и получения элементов.

Реализуйте аналогичную функциональность в C с использованием указателей и динамического выделения памяти.

Задание:

Сравните, как шаблоны упрощают код в C++ и как сложно реализовать аналогичное решение в C.

Задание 6: Исключения и обработка ошибок

Цель: Изучить механизмы обработки ошибок в C и C++.

Описание:

В C++ создайте класс с методом, который будет выбрасывать исключение в случае ошибки (например, если происходит попытка снять средства больше, чем доступно на счете).

Реализуйте аналогичную функциональность в C с использованием кодов ошибок и обработки ошибок.

Задание:

Сравните подходы к обработке ошибок в C и C++, обсудите преимущества использования исключений в C++.