Теория:

Модуль 1

1. **Переменные, операторы и массивы**

   a. **Основные команды командной строки**
   `cd`, `ls` (опции `-l` и `-a`), `pwd`, `cp`, `mv`, `rm` (опция `-r`), `mkdir`, программа `top`, компилятор `gcc` (опции `-o`, `-std=c99` и `-std=c23`). Перенаправление вывода `>`. Возвращаемое значение функции `main`. Функция `exit` из библиотеки `stdlib.h`.

   b. **Стандартный ввод/вывод**
   Ввод и вывод в языке C. Функции `printf` и `scanf` из библиотеки `stdio.h`. Что делают следующие спецификаторы: `%i`, `%2i`, `%02i`, `%p`.

   c. **Переменные**
   Что такое переменные? Переменные типа `int`. Инициализация переменных. Инициализация переменных. Чем инициализируются локальные переменные по умолчанию? Инициализация и присваивание, в чём различие между ними. Размер переменной типа `int`. Максимальное и минимальное значение для переменных типа `int`.

   d. **Операторы**
   Арифметические операторы(`+ - * / %`). Что делает оператор деления, если аргументы целочисленные и если аргументы – числа с плавающей точкой? Оператор присваивания (`=`). Оператор присваивания сложения и подобные ему (`+= -= *= /= %=`). Операторы инкремента и декремента (`++ --`). Префиксный и постфиксный инкремент/декремент, чем они отличаются. Операторы сравнения (`== != > < >= <=`). Что возвращают операторы сравнения? Логические операторы (`! || &&`). Оператор нахождения адреса (`&`). Оператор нахождения размера переменной (`sizeof`). Приоритет операторов.

   e. **Ветвление и циклы**
   Оператор ветвления `if-else`. Использования логических операторов в условии оператора ветвления. Цикл `while`. Цикл `for`. Цикл `do while`. Операторы `break` и `continue`.

   f. **Создание и инициализация массивов**
   Массивы. Элемент массива и индекс массива. Как хранятся массивы в памяти? Объявление и определение массивов. Инициализация массивов. Можно ли присваивать массив другому массиву с помощью оператора присваивания? Как распечатать массив? Размер массивов. Как узнать размер массива, используя оператор `sizeof`?

   g. **Двумерные массивы**
   Объявление, определение и инициализация двумерного массива. Как двумерный массив хранится в памяти?

   h. **Виды ошибок**
   Ошибки компиляции. Ошибки линковки. Ошибки времени выполнения. Логические ошибки. Неопределённое поведение.

2. **Функции и типы данных**

   a. **Основы работы с функциями**
   Параметры и аргументы функции. Возвращаемое значение функции. Объявления функции. Прототип функции. Определение функции. Возврат из функции. Ключевое слово `return`. Ключевое слово `void`.

   b. **Передача в функцию**
   Как переменные базовых типов и структуры передаются в функции? Как массивы передаются в функции? Три типа передачи аргументов в функцию: по значению, через указатель, через константный указатель. Передача одномерных и многомерных массивов в функции.

   c. **Рекурсия**
   Рекурсия. Алгоритмы вычисления факториала и алгоритм вычисления чисел Фибоначчи.

   d. **Целочисленные типы данных**
   Типы целочисленных переменных: `char`, `short`, `int`, `long`, `long long` и их `unsigned`-аналоги. Типичные размеры этих типов на современных системах и диапазоны значений, которые могут принимать данные типы. Создание новых названий для типов с помощью ключевого слова `typedef`. Что такое тип `size_t`. Когда он используется? Типы фиксированной ширины: `int8_t`, `uint8_t`, `int16_t` и другие. Целочисленное переполнение. Неопределённое поведение при целочисленном переполнении.

e. **Константы**
Квалификатор типа `const`. Разница между определение константы с помощью директивы `#define` и квалификатора `const`.

f. **Типы чисел с плавающей точкой**
Типы чисел с плавающей точкой: `float`, `double`. Типичные размеры этих типов на современных системах и диапазоны значений, которые могут принимать данные типы.

g. **Приведение типов.**
Неявное приведение типов. Когда оно происходит? Явное приведение типов. Как привести один тип в другой?

h. **Математическая библиотека** `math.h`
Функции `sqrt`, `exp`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `fabs`, `floor`, `log`, `pow`. Сравнение двух чисел с плавающей точкой с помощью функции `fabs`.

3. **Строки и текстовые файлы**

a. **Символы**
Кодировка ASCII. Использование целочисленного типа `char` для хранения кодов символов. Чтение и запись символов (спецификатор `%c`). Символьные литералы. Библиотека `ctype.h`. Функции `isalpha`, `isdigit`. Какие символы называются пробельными? Функция `isspace`. Функции `toupper` и `tolower`.

b. **Строки в языке C**
Что такое строка в языке C? Как строки хранятся в языке C. Символ завершения строки. Чтение и запись строк (спецификатор `%s`).

c. **Библиотека** `string.h`
Функции `strlen`, `strcpy`, `strcmp`. Функции `sprintf` и `sscanf`, использование этих функций для конвертации числа в строку и наоборот. Почему функция `strcpy` считается небезопасной?

d. **Аргументы командной строки**
Аргументы функции `main`: `argc` и `argv`. Что они означают и как их использовать?

e. **Текстовые файлы**
Открытие и закрытие файла, функции `fopen` и `fclose`. Режимы открытия файла `"w"` и `"r"`. Запись/чтение с помощью функций `fprintf` и `fscanf`. Запись/чтение по одному байту с помощью функций `fputc` и `fgetc`. Объект какого типа возвращает функция `fgetc` и почему? Что функция `fgetc` возращает при достижении конца файла? Глобальные потоки `stdin` и `stdout`.

4. **Указатели и структуры**

a. **Указатели**
Что хранят указатели? Объявление указателя. Инициализация указателя. Размер указателя на 32-х и 64-х битных системах. Операция нахождения адреса (`&`) переменной. Операция разыменования `*` указателя.

b. **Арифметика указателей**
Операторы, применимые ко всем указателям: оператор разыменования, операторы проверки на равенство/неравенство. Операторы, применимые к указателям, указывающим на элемент массива: `++`, `-`, прибавление/вычитание целого числа, вычитание двух указателей, оператор индексирования. Связь оператора индексирования (`[]`) с оператором разыменования. Когда использование этих операторов приводит к ошибке?

c. **Указатели и функции**
Передача переменной в функцию по указателю. Возврат нескольких объектов из функции с помощью параметров-указателей. Как передаётся массив в функцию? Array to pointer decay. Как передавать строки в функции. Как вернуть массив из функции? Возврат указателя из функции. Висячие указатели.

d. **Указатели разных типов**
Чем различаются указатели разных типов? Приведение типов указателей. Указатель `void*`. Константный указатель (`const int* p`) и постоянный указатель (`int* const p`).

e. **Структуры**
Объявление структуры. Определение структуры. Инициализация структуры. Поля структуры. Операции применимые к структурам. Доступ к полю структуры. Оператор точка. Присваивание структур. Оператор присваивания. Массив структур. Инициализация массива структур. Использование ключевого слова `typedef` для создания более короткого имени для структуры.

f. **Указатели на структуры**
Доступ к полю по указателю на структуру. Оператор стрелочка `->`.

g. **Передача структур в функции**
Три типа передачи в функции: передача по значению, передача по указателю и передача по константному указателю. В каких ситуациях нужно использовать тот или иной способ? Возврат структур из функций.

h. **Выравнивание**
Размер структуры. Выравнивание полей структуры. Оператор `alignof`.

5. **Сегменты**

a. **Сегменты памяти**
Что такое сегменты памяти? Ошибка Segmentation Fault.

b. **Сегмент памяти стек**
Сегмент памяти стек. Выделение и освобождение памяти в стеке. Размер стека. Реализация вызова функций с помощью сегмента памяти стек. Стековый кадр, что в нём хранится. Как можно переполнить стек? Переполнение стека при рекурсии.

c. **Сегмент памяти куча**
Динамическое выделение и освобождение памяти в куче: `malloc`, `realloc` и `free`. Преимущества и недостатки кучи перед стеком. Как обрабатывать случай, когда у `malloc` не получилось выделить запрашиваемое количество памяти? Утечки памяти. Программа `valgrind`. Повторное освобождение той же памяти. Возрат массива из функции с использованием выделения памяти в куче.

d. **Двумерный массив в сегменте куча**
Двумерный массив в сегменте куча. Два способа хранения такого массива: в виде одномерного массива и в виде массива указателей, каждый из которых будет указывать на соответсвующую строку массива.

e. **Сегмент памяти данные**
Сегменты памяти данные. Преимущества и недостатки сегмента данные перед стеком и кучей. Что такое глобальные переменные? Что такое статические переменные? Когда и как инициализируются глобальные и статические переменные? Строковые литералы. Где хранятся строковые литералы?

f. **Сегмент памяти Текст**
Сегмент памяти текст. Преобразование кода программы в код на языке ассемблера и в двоичный код. Указатели на функции. Объявление указателей на функции. Передача указателей на функции в другие функции. Стандартная функция `qsort` и передача в ней компаратора.

6. **Динамический массив**

a. **Виды массивов в языке C**
Массив в сегменте стек. VLA-массив. Массив в сегменте данные. Массив в сегменте куча. Преемущества и недостатки создания каждого этих массивов.

b. **Создание своего динамического массива**
Создание своего динамического массива на основе массива в куче. Поля такого массива: указатель на данные в куче, размер (`size`) и вместимость (`capacity`). Чем размер отличается от вместимости? Функции для работы с нашим динимическим массивом:

- `init` – инициализируем массив
- `destroy` – уничтожаем массив и освобождаем всю память
- `get` – возвращает `i`-й элемент
- `set` – устанавливаем `i`-й элемент
- `reserve` – изменяет вместимость динамического массива
- `resize` – изменяет размер динамического массива
- `push_back` – добавляет один элемент в конец динамического массива
- `pop_back` – удаляет последний элемент массива и возвращает его
- `shrink_to_fit` – делает вместимость массива равной её размеру

Поверхностная и глубокая копия динамического массива. Как изменить тип элемента, который хранится в динамическом массиве? Использование `typedef` для настроики типа элемента массива.

c. **Макросы-константы**
Директива препроцессора `#define`. Создание констант с помощью макросов. Условная компиляция. Директивы препроцессора `#if`, `#ifdef`, `#ifndef`, `#else`, `#elif`, `#endif` и оператор `defined`. Флаг `-D` компилятора gcc. Предопределённые макросы: `__FILE__`, `__LINE__`, `__DATE__`, `__TIME__`, `__VERSION__`, `__cplusplus`. Макросы для определение операционной системы: `_WIN32`, `_WIN64`, `__linux__`, `__APPLE__`.

d. **Заголовочные файлы**
Директива препроцессора `#include` и что конкретно она делает. Header-файлы. Стражи включения. `#pragma once`.

e. **Функциональные макросы**
Функциональные макросы (function-like macros). Многострочные макросы. Типичные ошибки, которые могут возникнуть при работе с функциональными макросами. Использование оператора `do-while` в многострочных функциональных макросах. Операция stringification (`#`). Операция concatenation (`##`). Макрос `assert` из библиотеки `assert.h`. Написание макроса, аналогичного макросу `assert`. Флаг `-E` компилятора gcc.

f. **Макросы и динамический массив**
Использование макросов, для генерации кода динамического массива заданного типа.

7. **Память и бинарные файлы**

a. **Шестнадцатеричная система счисления**
Шестнадцатеричная система счисления. Печать и считывания целочисленных переменных в шестнадцатеричной системе с помощью спецификатора `%x`.

b. **Представление чисел в памяти**
Представление целых положительных чисел в памяти. Представление целых отрицательных чисел в памяти. Дополнительный код. Представление чисел с плавающей точкой в памяти с использованием стандарта IEEE 754.

c. **Побитовые операторы**
Побитовые операторы ($\sim$ `& | ^ << >>`). Побитовые операторы присваивания (`&= |= ^= <<= >>=`). Получение отрицательного числа и помощью побитовых операций. Изменение i-го бита числа с помощью побитовых операций. Печать всех битов числа.

d. **Память**
Порядок байт. Little endian и big endian. Просмотр байт объекта с помощью указателя типа `unsigned char*`. Функции `memset`, `memcpy`, `memmove` из стандартной библиотеки.

e. `* (необязательный вопрос)` **Алиасинг**
Что такое алиасинг? Strict Aliasing Rule. Неопределённое поведение при алиасинге.

f. `* (необязательный вопрос)` **Бинарные файлы**
Бинарный и текстовый файл. Бинарный и текстовый режимы открытия файла. Как открыть файл в бинарном и текстовом режиме? В чём отличие бинарного и текстового режима в операционных системах Linux и Windows? Как хранится перенос строки в операционных системах Linux и Windows? Разница между CRLF и LF переносами строки. Запись из памяти в файл и чтение из файла в память с помощью функций `fwrite` и `fread`. Функции `fseek` и `ftell`. Использование функции `fgetc` для побайтового чтения файла. Программа `xxd` для просмотра байт файла.