Теория:

Модуль 1

1. **Основы**

   a. **Основные команды командной строки**
   `cd`, `ls` (опции `-l` и `-a`), `pwd`, `cp`, `mv`, `rm` (опция `-r`), `mkdir`, `find`(опция `-name`), программа `top`, компилятор `gcc` (опции `-o`, `-std=c99`, `-lm` и `-S`). 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`.

   c. **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`. Тернарный оператор. Чем отличается тернарный оператор от `if-else`?

   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 (`! || &&`). Побитовые операторы (∼ `& | ^ << >>`). Тернарный оператор (`? :`). Оператор нахождения адреса (`&`). Оператор нахождения размера переменной (`sizeof`). Оператор разыменования (`*`). Оператор обращения к элементу массива (`[]`) и его связь с оператором разыменования. Оператор доступа к полю структуры (`.`). Оператор доступа к полю структуры через указатель на структуру (`->`). Приоритет операторов.

2. **Массивы**

   a. **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?

   b. **Передача массивов в функции**
   Как передаются массивы в функции? Array to pointer decay. Как вернуть массив из функции?

   c. **Двумерные массивы**
   Объявление, определение и инициализация двумерного массива. Как двумерный массив хранится в памяти? Как двумерный массив передаётся внутрь функции?

   d. **Простейшие алгоритмы**
   Один простой алгоритм сортировки (выбором или пузырьком). Алгоритм перемножения матриц.

3. **Функции**

   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 передаюся 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, чисел Фибоначчи и бинарного возведения в степень с помощью рекурсии. Переполнение стека при рекурсии.

4. **Типы данных**

   a. **Создание переменных**
   Переменные. Понятия объявления, определения, в чём различие между ними. Инициализация и присваивание, в чём различие между ними.

   b. **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.

   c. **Представление целочисленных переменных в памяти**
   Как хранятся в памяти отрицательные числа? Дополнительный код. Целочисленное переполнение. Неопределённое поведение при целочисленном переполнении.

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

   e. **Типы чисел с плавающей точкой**
   Типы чисел с плавающей точкой: `float`, `double` и `long double`. Типичные размеры этих типов на современных системах и диапазоны значений, которые могут принимать данные типы. (*)Представление чисел с плавающей точкой в памяти. (*)Стандарт IEEE 754.

   f. **Приведение типов.**
   Неявное приведение типов. Когда оно происходит? Явное приведение типов, как привести один тип в другой.

   g. **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`.

5. **Строки**

   a. **Символы**
   Кодировка ASCII. Использование целочисленного типа `char` для хранения кодов символов. Чтение и запись символов (спецификатор `%c`).

   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.

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

6. **Указатели и структуры**

   a. **Указатели**
   Объявление указателя. Инициализация указателя. Размер указателя на 32-х и 64-х битных системах.

   b. **Арифметика указателей**
   Операторы, применимые к указателям и что они делают: `++`, `-`, прибавление целого числа, вычитание двух указателей, разыменование и оператор взятие индекса (`[]`). Операции нахождения адреса (`&`) переменной и операция разыменования `*` указателя.

   c. **Указатели разных типов.**
   Чем различаются указатели разных типов. Указатель `void*`. Константный указатель (`const int* p`) и постоянный указатель (`int* const p`). Передача в функцию переменных по указателю. Передача массивов в функции.

d. **Структуры**
Объявление структуры. Определение структуры. Инициализация структуры. Поля структуры. Доступ к полю структуры.

e. **Указатели на структуры**
Доступ к полю по указателю на структуру. Оператор `->`. Передача структур в функции и возврат их из функций.

f. **Выравнивание**
Размер структуры. Выравнивание полей структуры.

7. **Сегменты**

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

b. **Сегмент памяти стек**
Сегмент памяти стек(не путать с абстрактным типом данных – стек). Выделение и освобождение памяти в стеке. Размер стека. Реализация вызова функций с помощью сегмента памяти стек. Стековый кадр, что в нём хранится. Адрес возврата. Как можно переполнить стек?

c. **Сегмент памяти куча**
Динамическое выделение и освобождение памяти в куче: `malloc`, `calloc`, `realloc` и `free`. Преимущества и недостатки кучи перед стеком. Динамический массив, его создании с помощью функции `malloc`.

d. **Двумерный динамический массив**
Динамический двумерный массив, его создании с помощью функции `malloc`. Хранение двумерного массива в виде одномерного массива.

e. **Утечки**
Утечки памяти. Основы работы с valgrind.

f. **Сегмент памяти Данные**
Сегменты памяти data и rodata. Чем они различаются? Что такое глобальные переменные. Что такое статические переменные. Где хранятся глобальные и статические переменные? Когда и как инициализируются глобальные и статические переменные? Строковые литералы. Где хранятся строковые литералы?

g. **Сегмент памяти Текст**
Сегмент памяти text. 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.

8. **Файлы**

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

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

c. **Алиасинг**
Что такое алиасинг? Strict Aliasing Rule. Неопределённое поведение при алиасинге.

d. **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`. Что возвращает функция `fgetc`?

e. **Бинарные файлы**
Бинарный и текстовый режимы открытия файла, в чём их отличие. Как хранится перенос строки на операционных системах Linux и Windows. Запись из памяти в файл и чтение из файла в память с помощью функций `fwrite` и `fread`. Функции `fseek` и `ftell`.

9. **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

Как изменить тип элемента, который хранится в динамическом массиве?

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__`, `__ANDROID__`.

d. **Основы раздельной компиляции**
Директива препроцессора `#include` и что конкретно она делает. Header-файлы. Стражи включения. `#pragma once`. Раздельная компиляция. Разделение кода программы на объявления и определения. Стадии сборки программы (препроцессинг, компиляция, линковка). Флаг `-c` компилятора gcc.

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.

Дополнительные материалы для подготовки:

1. Керниган Ритчи Язык программирования С.
   lib.mipt.ru/book/266005/

2. Язык С и структуры данных в Йелле.
   www.cs.yale.edu/homes/aspnes/classes/223/notes.html