Домашнее задание №1
Итоги

Общие проблемы

- Имя пользователя на GitHub и название репозитория (указания разосланы)
- README.md
- Рабочий язык — английский
- Оформление кода
- Самодостаточные заголовки
- Мейкфайлы
- using namespace std
- Передача по ссылке или по значению?
- Считывание строки из 2+ слов
- Видимость переменных

README.md

```
# Assignment 1
## Author
Иванов Иван Иванович, группа 24.Б84-мм
## Contacts
stXXXXXX@stdudent.spbu.ru
## Description
Home assingment 1 - extended "Hello world". First says "Hello,
World". After this reads a string in a cycle and says "Hell,
<string>"
## Build
make
## Run
./hello
```

• Рабочий язык — английский

- В коде и репозитории используется только английский язык
- Описание задач и лабораторных на английском языка
- Не знаете как выразить мысль на английском - воспользуйтесь машинным переводом или попросите помощи у преподавателя английского языка
- Исключения:
  - Фамилия Имя Отчество и номер группы в README.md
  - Общение в Issues и в Pull requests

Оформление кода

- Обязательный комментарий в начале каждого файла исходно кода и заголовка
- Поможет проверяющему не запутаться в открытых файлах и ускорит проверку
- Отступы 4 пробелами (не табуляцией)
- Фигурные скобки на пустых строках
- Понятные идентификаторы

Самодостаточные заголовки

- Заголовок нужно оформлять так чтобы его можно было включить в любой файл исходного кода сколько угодно раз не внося в исходный код правки кроме директив `#include “myheader.h“`
- Не забывайте про include guard
- Убедитесь, что объявили (сами или включением другого заголовка) все что нужно в начале заголовка
- Не объявляйте и не подключайте ничего лишнего

Мейкфайлы

- Называйте мейкфайл “Makefile“ (с заглавной буквы без расширения)
- Каждый файл исходного кода компилируйте в объектный файл отдельной целью
- Прописывайте в зависимости заголовочные файлы
- Убедитесь, что цель на очистку есть и работает корректно
- Лучше разделить цели clean (удалить объектные файлы) и cleanall (удалить объектные файлы и результаты сборки — исполнимые программы и библиотеки)

using namespace std

- Не используйте `using namespace std`
- Особенно если хотите перегрузить что-то из `std`
- Использование в заголовке - серьёзная ошибка
- А лучше вообще не используйте (по крайне мере первые полгода-год)

• Передача по ссылке или по значению?

- Ссылка позволяет безопасно получить доступ к существующему объекту
- Ссылки позволяют избежать копирования крупных объектов при передаче в функции
- Защита от повреждения — константная ссылка

```
void sayHi(const std::string &name);
```

- Иногда все же предпочтительно копировать — но это достаточно специальные случаи

Считывание строки из 2+ слов

```
std::string in;

std::cin >> in; //Will read till whitespace or EOL

std::getline(std::cin, in); //Will read till EOL
```

Видимость переменных

- Как только можно сужать область видимости переменных
- Исключить на 99,9% использование глобальных переменных

Домашнее задание

- Два упражнения по использованию памяти и файлов (Assignment2a, Assignment2b)
- Assignment2a
  - Открыть бинарный фал
  - Узнать его размер
  - Выделить под него массив
  - Считать файл в массив
  - Развернуть массив a[i]←a[N-i]; a[N-i]← a[i]
  - Записать в новый файл одной командой
- Assignment2a
  - Считать строку из чисел (операндов) и знаков операций (+-*/), элементы разделены пробелами
  - Интерпретировать строку как обратную польскую запись выражения и вычислить это выражение

Assignment2a

- Узнать размер файла можно с помощью `std::filesystem::file_size`
- Чтение и запись файлов

```
std::ifstream infile;
infile.open("source.pdf",std::ios::binary|std::ios::in);

std::ofstream outfile;
```

-
```
std::outfile.open("temppdf.pdf",std::ios::binary|std::ios::out);
infile.read((char *)&buffer,sizeof(buffer))
outfile.write((char *)&buffer,sizeof(buffer));

infile.close();
outfile.close();
```

- Выделение памяти: `new[]`
- Не забудьте вызвать `delete[]`

Assignment2b

- Стек: first input last output
- Алгоритм вычисления
- Обработка входного символа
  - Если на вход подан операнд, он помещается на вершину стека.
  - Если на вход подан знак операции, то соответствующая операция выполняется над требуемым количеством значений, извлечённых из стека, взятых в порядке добавления. Результат выполненной операции кладётся на вершину стека.
- Если входной набор символов обработан не полностью, перейти к шагу 1.
- После полной обработки входного набора символов результат вычисления выражения лежит на вершине стека.
- Стек сделать в виде массива (new[]-delete[])
- На вершину стека «смотрит» указатель
- Работающий пример — Расширенный эмулятор МК 61/54 для Android by Stanislav Borutsky https://apkpure.com/ru/mk-61-54/com.cax.pmk