О ПРАКТИКЕ СТУДЕНТОВ 2 СЕМЕСТРА

1. **Общие сведения**
   1.1. Задание на практику выдается студентам на второй семестр и предполагает завершение его самостоятельного выполнения студентами не позднее начала экзаменационной сессии.
   1.2. Один и тот же вариант задания выдается, как правило, двум студентам в одной группе и может выполняться ими как независимо, так и во взаимодействии. В последнем случае двумя студентами сдаются два фактически идентичных варианта отчета, отличающиеся лишь титульными листами с фамилией исполнителя (и аналогично для бланков заданий).
   1.3. Получение оценки за практику помимо сдачи отчета предполагает его защиту студентами.

2. **Структура задания**
   2.1. Задание формулируется в текстовом виде, содержит краткое описание постановки задачи, перечень исходных данных и формат их преставления, а также формат сохранения результатов.
   2.2. Как правило, задания относятся к ранее изученным курсам «Аналитическая геометрия» и «Математический анализ», хотя могут и несколько выходить за их рамки, и требуют программной реализации алгоритма решения конкретной задачи.
   2.3. Конкретные алгоритмы решения в заданиях как правило не указаны; предполагается, что они должны быть разработаны студентами самостоятельно. При необходимости студентам рекомендуется обращаться к необходимым дополнительным источникам информации, как печатным, так и электронным.
   2.4. Задание предполагает реализацию алгоритма решения задачи на языке программирования C++. Эффективность алгоритма, характеризуемая временем, затрачиваемым на его исполнение, не является критичной, если время счета не превышает нескольких секунд. Качество написанного кода будет учитываться при выставлении оценки по итогам защиты отчета.

3. **Подготовка отчета и процедура его защиты**
   3.1. По окончании работы над заданием студент (или пара студентов, работающих совместно) после завершения подготовки отчета по практике до его защиты должен обратиться к преподавателю, ответственному за практику, который, в свою очередь, может выдать ему тестовое задание. Если таковое выдано и разработанные студентом алгоритмы позволяют решить тестовую задачу верно, то студент получает допуск к защите отчета. Возможно, к середине или концу весеннего семестра будет реализована система автоматизированной проверки правильности реализации алгоритмов.

3.2. Отчет оформляется в MS Word. Качество оформления отчета учитывается при выставлении оценки за практику.

3.3. Основная часть отчета должна содержать:

- рисунок или схему, отражающую постановку задачи (ее можно нарисовать как в графических пакетах, так и непосредственно в MS Word)[1];
- описание метода или методов решения задачи с приведением всех необходимых для выполнения расчетов формул; иллюстрации, способствующие восприятию материала, приветствуются, однако избыточным количество рисунков быть не должно;
- описание алгоритма, которое может сопровождаться блок-схемой; если блок-схемы достаточно для понимания работы алгоритма, можно ограничиться ей;
- обсуждение особенностей реализации алгоритма, разбор частных случаев и т.п.;
- примеры решения задач с проверкой правильности получаемых результатов.

Тексты программ включать в отчет **не нужно**.

3.4. В ходе выполнения задания настоятельно рекомендуется провести анализ сложности реализуемых алгоритмов: как правило, наиболее информативными являются сведения о количестве выполняемых арифметических операций. Эти сведения целесообразно включить в отчет.

3.5. Во время защиты отчета по практике студентами демонстрируются презентации небольшого объема, рассчитанные на 5-7 минут, в которых представляются основные полученные результаты. В ходе защиты студенту (или паре студентов, работающих совместно) обязательно будут заданы вопросы, качество ответов на которые с учетом качества подготовленных отчетов, презентации, а также кода программы определяет оценку за практику.

4. Программная реализация

4.1. Программная реализация алгоритма решения задачи выполняется на языке программирования C++.

4.2. Программа должна быть написана таким образом, чтобы она была кросс-платформенной: при автоматизированной проверке правильности решения задач Ваш код будет автоматически скомпилирован в операционной системе Linux компилятором `g++`. Вы, разумеется, можете разрабатывать код в любой операционной системе, используя любую удобную Вам среду программирования.

---

[1] В 2024-25 учебном году такого требования явно не формулировалось; в представленных примерах отчетов таких иллюстраций нет, что усложняет их восприятие материала

4.3. Программа может состоять как из одного файла с расширением `.cpp`, так и из нескольких файлов с расширениями `.cpp`, `.h`, `.hpp`. При этом предполагается, что все файлы кода находятся в одном каталоге (рис. 1).

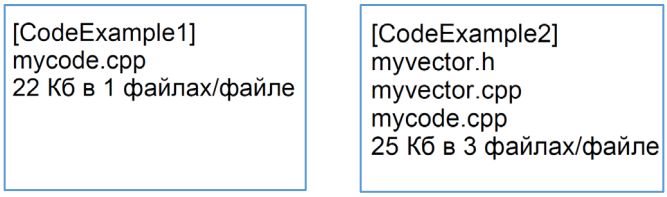

Рис. 1. Возможные способы организации кода основной программы

4.4. Обратите внимание, что Linux – регистрочувствительная операционная система, поэтому, например, файлы с именами “`abc.cpp`”, “`Abc.cpp`”, “`ABC.cpp`”, “`abc.Cpp`” и т.д. являются разными и вполне могут присутствовать в одной папке. Это необходимо учитывать, в частности, при подключении библиотек после директивы `#include` в коде программы: писать имена файлов нужно в правильном регистре. Также в Linux при разделении имен каталогов нужно использовать значок «слэш» («/»), например,

```
std::ofstream outFile("myvector/myvector.h");
```

При этом в Windows допустимы и «слэш», и «обратный слэш» (т.е. «/» и «\»), поэтому для обеспечения кросс-платформенности следует использовать «слэш» («/»).

4.5. В систему автоматизированного тестирования (если она будет готова к моменту защиты) Вы будете отправлять исходный код Вашей программы, а не файлы с результатами расчетов.

4.6. Исходные данные должны поступать в стандартный поток ввода, т.е. при исполнении программы они должны считываться из потока `std::cin`. Чтобы облегчить процедуру отладки и не вводить многократно одни и те же данные каждый раз с клавиатуры, рекомендуется использовать следующий прием: ввести необходимые данные в текстовый файл так, как это указано в задании, сохранив его под произвольным именем, например, `input.txt`, а затем при запуске программы вместо исполнения, к примеру

```
mycode.exe
```

исполнять команду

```
mycode.exe < input.txt
```

Если Вы работаете в MS Visual Studio, то для автоматизации этого действия на этапе отладки кода зайдите в меню Проект –> Свойства проекта, в открывшемся окне в левом столбце выберите «Отладка», и в строке «Аргументы команды» укажите нужную конструкцию (рис. 2).

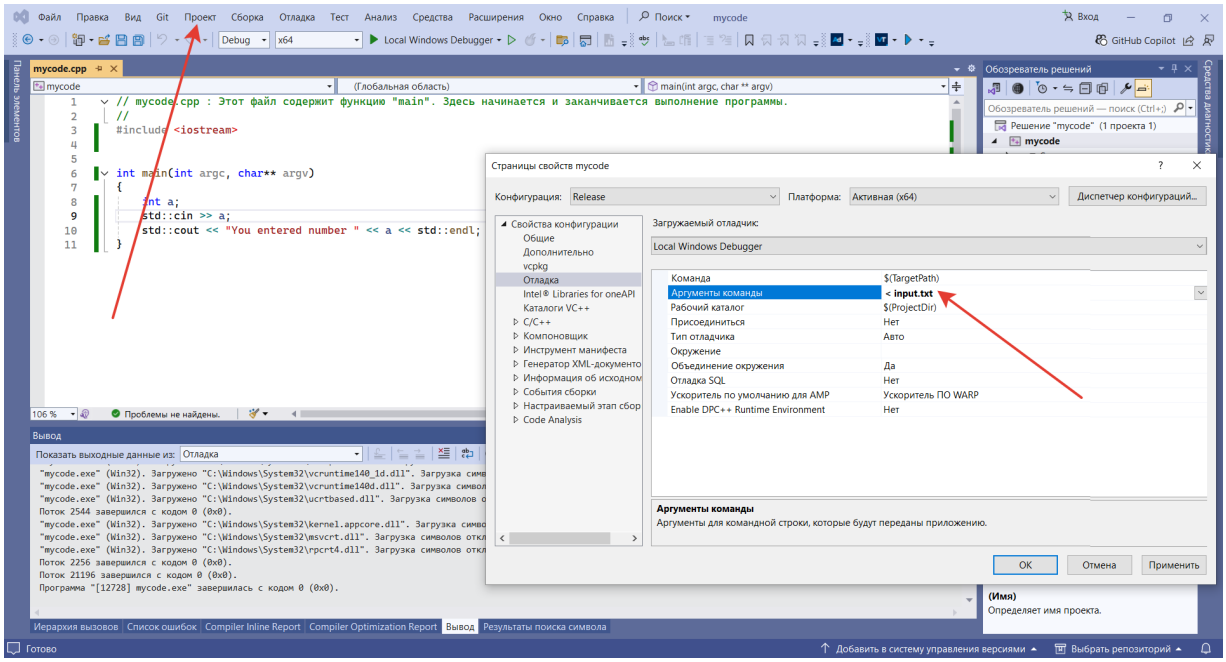

Рис. 2. Настройка параметров командной строки в MS Visual Studio

4.7. Результаты вычислений необходимо вывести в стандартный поток вывода `std::cout` в том формате, что указан в задании. При желании сохранить результаты вывод можно направить в файл, исполнив команду

```
mycode.exe > output.txt
```

Если исполнить код

```
mycode.exe < input.txt > output.txt
```

то исходные данные будут считаны из файла `input.txt` в текущем каталоге, а вывод вместо экрана будет перенаправлен в файл `output.txt`

Похожий механизм будет использован в системе автоматизированной проверки кодов.

4.8. Поскольку основной объем расчетов Вы будете выполнять с вещественными числами, работа с коротыми приводит к появлению погрешности, проверка правильности получаемых результатов будет производиться следующим образом:

- если должно получиться число x, не равное нулю, а Ваша программа выводит значение y, то ответ будет засчитан верным, если относительная погрешность не превысит 10^{-6}, т.е.

$$\frac{|x-y|}{|x|} < 10^{-6};$$

- если же «эталонное» значение x равно нулю, то ответ верен, если $|y| < 10^{-6}$.

Теоретически, такую точность может обеспечивать тип данных `float`, однако настоятельно рекомендуется все вычисления производить с двойной точностью (тип данных `double`). Целочисленные значения погрешности не допускают.