Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Изисквания за проектите по ООП и ООП-практикум

(летен семестър 2021/2022 г.)

Проектът е цялостна задача, която трябва да решите с помощта на познанията по C++, получени през летния семестър. Изискванията за проектите по ООП и ООП-практикум са следните:

1. По време на работата по проекта трябва да се използва хранилище в системата Github.
    - За създаването на хранилището можете да поискате помощ от вашите асистенти.
    - Хранилището трябва да се обновява регулярно, паралелно с работата ви по проекта.
    - Направените от вас междинни промени няма да бъдат оценявани, ще се оценява само крайния резултат, но могат да бъдат използвани за проверка при съмнение за плагиатство.
2. В обявения срок трябва да предадете:
    - документация на проекта;
    - изходен код на решението;
    - няколко примера, подбрани от Вас, които демонстрират работата на задачата.
3. Предаването става чрез прикачване на ZIP архив към съответното задание в Moodle, който съдържа всички файлове, необходими за компилирането на проекта и документацията към проекта.
    - Включването на допълнителни библиотеки е препоръчително да става с използването на Git модули. За повече информация можете да се обърнете към преподавателския екип.
4. Препоръчително е документацията на проекта да съдържа:
    - анализ на задачата и Вашия подход за решение (на какви стъпки сте разделили решението, какъв метод или алгоритъм сте избрали, как сте се справили с конкретен проблем);
    - кратко описание на класовете, създадени от Вас, за решение на задачата (избраната архитектура, описание на член-данните и член-функциите на класовете и начина им на използване);
    - идеи за бъдещи подобрения;
    - връзка към вашето хранилище в Github (задължително).
    - Примерно съдържание на документацията е описано по-долу.

- Препоръчваме на студентите да разучат някоя система за генериране на документация от коментари в изходния код, като Doxygen или всяка друга аналогична система. При използване на такава система отпада изискването да се предава текстовата документация, описана по-горе, но се изисква описание в коментарите на кода на създадените класове, както и на техните методи и член-данни. Като част от курсовия проект трябва да се предадат и генерираните файлове с документация.

5. По време на защитата трябва в рамките на 10 минути да представите Вашето решение и да демонстрирате работата на програмата с подготвени от Вас данни.
6. По време на защитата се очаква да можете да отговорите на различни въпроси, като например:
   - каква архитектура сте избрали;
   - защо сте избрали именно нея;
   - дали сте обмислили други варианти и ако да – кои;
   - как точно работят различните части от вашия код и какво се случва на по-ниско ниво;
   - други въпроси относно решението и избраните средства за реализацията му.
7. Възможно е да Ви бъде поставена малка задача за допълнение или промяна на функционалността на проекта ви, която вие трябва да реализирате за максимум 1 час.
8. Невъзможност да реализирате малката задача на място означава, че не познавате добре проекта си и поражда съмнения, че сте ползвали чужда помощ за реализацията му. Последното ще се отрази негативно на оценката Ви.
9. Установено плагиатство на код от колеги и от други източници води до анулиране на работата и оценка Слаб 2 за курсовете по ООП и ООП-практикум. **За плагиатство се счита използване в решението на код, чиито източник не е изрично упоменат.**
10. **Критерии за оценка на проекта**
    Проектите се оценяват по редица от критерии, част от които са описани по-долу.
    Тъй като курсът се фокусира върху обектно-ориентираното програмиране и неговата реализация в езика C++, най-важното изискване за проектите е те да са изградени съгласно добрите принципи на ООП. Решението, в което кодът е процедурен, има лоша ООП архитектура и т.н. се оценява с оценка Слаб 2. Други важни критерии за оценка на проектите са:
    - Дали решението работи коректно, съгласно спецификацията. Решение, което не работи и/или не се компилира носи ниска или слаба оценка.
    - Дали решението отговаря на заданието на проекта.
    - Дали е извършен анализ на поставената задача и на различните подходи за нейното решаване.
    - Каква част от необходимата функционалност е била реализирана.
    - Дали са обхванати всички потенциални сценарии за използване на функционалността.
    - Дали е постигната оптимална сложност на реализираните алгоритми (ако има такива и спрямо текущите ви познания).

- Дали е постигнато оптимално управление на паметта, включително заделяне и освобождаване на динамична памет, статични член-данни, глобални променливи.
- Оформление на решението. Проверява се дали кодът е добре оформен, дали е спазена конвенция за именуване на променливите, дали е добре коментиран и т.н.
- Дали решението е било добре тествано. Проверява се какви тестове са били проведени върху приложението, за да се провери дали то работи коректно. Очаква се по време на защитата да можете да посочите как сте тествали приложението, за да проверите дали то работи коректно и как се държи в различни ситуации.
- Идеи и потенциал за бъдещи разширения.
- Представяне на проекта.
- Възможност за бърза реализация на малка промяна в проекта (познаване на кода) и отговаряне на въпроси.

11. Оценката на всеки от проектите се формира от онази негова част, която е била самостоятелно разработена от Вас. Допустимо е да използвате код написан от някой друг (напр. готова библиотека или помощ от ваш асистент), но (1) той не носи точки към проекта и (2) това трябва да бъде ясно обявено както при предаването, така и при защитата на проекта, като ясно обозначите коя част от проекта сте разработили самостоятелно. Това означава, че:

    - Използваният наготово код трябва да се маркира ясно, като поставите коментари на подходящи места в кода си.
    - По време на защитата трябва да посочите кои части сте разработили самостоятелно и кои са взети от други източници.

12. Както е написано по-горе, когато в проекта си използваме чужд код, сам по себе си той не ви носи точки. Допълнителни точки могат да се дадат или отнемат, според (1) способността ви за внедряване на кода във вашето решение (напр. в случаите, когато се използва външна библиотека) и за това (2) дали добре разбирате какво прави той.

Примерна структура на документацията

Съдържание (препоръчителен обем без приложенията: от 3 до 5 стр.)

Глава 1. Увод (1 стр.)

1.1. Описание и идея на проекта (3–4 изр.)

1.2. Цел и задачи на разработката (½–1 стр.)

1.3. Структура на документацията (3–4 изр.)

Глава 2. Преглед на предметната област (½–1 стр.)

2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

2.2. Дефиниране на проблеми и сложност на поставената задача

2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите

2.4 Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)

Глава 3. Проектиране (1–2 стр.)

5.1. Обща архитектура – ООП дизайн

5.3. Диаграми (на структура и поведение – по обекти, слоеве с най-важните извадки от кода)

Глава 4. Реализация, тестване (1–2 стр.)

4.1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

4.2. Управление на паметта и алгоритми. Оптимизации.

4.3. Планиране, описание и създаване на тестови сценарии (създаване на примери)

Глава 5. Заключение (2–3 изр.)

5.1. Обобщение на изпълнението на началните цели

5.2. Насоки за бъдещо развитие и усъвършенстване

Използвана литература

Препоръка за оформяне на документацията на проекта:

1. Шаблонът е препоръчителен и може да се променя в зависимост от конкретното задание.
2. Йерархията на структуриране на съдържанието да не бъде повече от 3 нива, номерирани с арабски цифри – напр. 1.2.3.
3. Чуждестранните термини да бъдат преведени, а където това не е възможно – цитирани в *курсив* и нечленувани.
4. Страниците да бъдат номерирани с арабски цифри, в долния десен ъгъл.
5. Използваният шрифт за основния текст на описанието да бъде Times New Roman 12 или Arial 10, и Consolas за кода.
6. Да се избягва пренасянето на нова страница на заглавия на секции, фигури и таблици.
7. Да се избягват празни участъци на страници вследствие пренасянето на фигури на нова страница.
8. Всички фигури и таблици да бъдат номерирани и именувани (непосредствено след фигурата или таблицата).
9. Всички фигури и таблици да бъдат цитирани в текста.
10. Всеки термин, дефиниция, алгоритъм или информация, която е взета от литературен източник или Интернет трябва да бъде цитирана.
11. Всички цитати да бъдат отразени в списъка на използваната литература.

12. Всички източници от списъка на използваната литература да бъдат цитирани в текста.
13. Използваната литература да се цитира съгласно MLA Style