Правила

Следните правила описват процеса по заявяване и защитаване на курсови проекти по Увод в програмирането.

- Срок за заявяване на проекти: 01.12.2013 (за всички силози). Заявяването става чрез…. (има квота за максимален брой студенти, работещи по еднакъв проект: 5)
- По проектите се работи самостоятелно (т.е. не се допуска работа в екипи)
- Плагиатство от колеги и от други източници води до анулиране на работата. Това не означава, че код не може да се взаимства, но всеки взаимстван фрагмент код трябва да се разбира отлично, да може да се модифицира и обяснява
- Срок за предаване: до 6 седмици след датата на заявяване на проектите (чрез прикачване на файл към съответното задание в Moodle)

Проекти по УП

- Силоз 1: Игри
  - Snake
  - Alien Attack
  - Sokoban
  - Tetris
  - Breakout
  - Xonix
  - Pacman
  - Mine Sweeper
  - Pong
  - Lander
  - Arcade Voleyball
  - Frogger
  - The Game of Life
- Силоз 2: Изчислителни
  - Матрици и вектори
  - Големи числа
  - Геометрия
  - Дати
  - Кодирания RLE и Base64
  - XML Parser

Силоз 1: Игри

Snake

Примерна игра: http://html5games.com/2012/09/snake-game/

Alien Attack

Примерна игра: http://html5games.com/2012/10/space-invectors/

Sokoban

Примерна игра: http://html5games.com/2011/03/caravan-sokoban/

Tetris

Примерна игра: http://html5games.com/2013/06/tetris-9-cells/

Breakout

Примерна игра: http://html5games.com/2011/05/html5-breakout/

Xonix

Примерна игра: http://www.youtube.com/watch?v=POhMfAFZ_6c

Pacman

Примерна игра: http://html5games.com/2012/08/html5-pacman/

Mine Sweeper

Примерна игра: http://html5games.com/2010/07/mine-sweeper/

Pong

Примерна игра: http://ricklamers.nl/pong/

Lander

Примерна игра: http://html5games.com/2010/09/jslander/

Arcade Voleyball

Примерна игра: http://html5games.com/2010/04/blobby-volley-2/

Frogger

Примерна игра: http://html5games.com/2011/04/turtle-rescue/

The Game of Life

(http://en.wikipedia.org/wiki/Conway's_Game_of_Life
http://bg.wikipedia.org/wiki/%D0%96%D0%B8%D0%B2%D0%BE%D1%82_(%D0%B8%D0%B3%D1%80%D0%B0)

Да се напише програма, която реализира “Игра на Живот”. В Играта на Живот има "дъска" състояща се от NxM клетки (10<N<60,10<M<60, дори да е възможна с по-малки стойности от дадените играта е безинтересна). Във всяка клетка или има живо същество или не. Живите същества могат да бъдат от различни племена (т.е условно можем да ги наречем Варвари, Рицари, Граждани и т.н). Всяко племе си има собствен “знак” (т.е ‘%’ за Варварин, “+” за Рицар, “&” за Гражданин и т.н). Дъската има начално състояние. То определя се в кои клетки има живот, в кои не и кое същество от кое племе е. Играта се състои от безброй много ходове, като на всеки ход дъската се променя по следните правила:

- Всяка жива клетка с по-малко от две живи(независимо от племето) съседни клетки умира (от самота).
- Всяка жива клетка с повече от три живи (независимо от племето) съседни клетки умира (от пренаселеност).
- Всяка жива клетка с две или три живи(от същото племе) съседни клетки остава жива и на следващата итерация.
- Всяка мъртва клетка с точно три живи (от същото племе) съседни клетки се превръща в жива клетка (от съответното племе).
- * Ако за дадена "мъртва" клетка се удовлетворява условието за раждане за N племена, "съдбата" (т.е остатъка от целочислено деление на random()/N) решава от кое племе да бъде тя.

След края на всеки ход дъската трябва да се визуализира на екрана. Между два последователни хода трябва да се изчака известно време за да може човек да наблюдава развитието на играта.

Бонус:

- Всеки път дъската се инициализира с произволен размер, брой племена и състояние на клетките.
- Добавете опция за игра до “победа”. Като (някои) от избираемите условия за победа могат да са:
    - Да остане само едно “живо” племе
    - Да умрат всички племена
    - Дъската да не се е променила в рамките на два последователни хода

Силоз 2: Изчислителни

Матрици и вектори

Да се напише програма, която позволява операции с матрици и вектори от дробни (рационални) числа:

1. Матрици

- въвеждане на матрица от клавиатурата
    - да се задават размерности и да се въвеждат елементите по удобен начин
- извеждане на матрица на екрана
    - да се извежда матрицата в подреден табличен вид
- събиране и изваждане на матрици
    - да могат да се правят няколко събирания или изваждания последователно, например A + B + C + D
- умножение на матрици
    - да се прави проверка дали размерностите на матриците са подходящи
- транспониране на произволна правоъгълна матрица
- намиране на детерминанта на матрица
    - без значение кой алгоритъм ще бъде използван
- намиране на обратна матрица
    - да се извежда грешка, ако матрицата няма обратна

2. Вектори

Разглеждаме векторите като матрици с 1 ред или 1 колона (по ваш избор). Да се реализират следните операции за работа с вектори:

- скаларно и векторно произведение на вектори

- нормализиране на вектор
- умножение на вектор с матрица (линейна траснформация)

Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на горните операции.

Бонуси:
- персистентност (работа с файл)
- редактиране на матрица с курсор (както в електронна таблица)
- реализиране на именувани матрични променливи

Големи числа

“Големите числа” са цели положителни числа, представени, без практическо ограничение за тяхната големина (освен наличната памет). Ако би представялвало улеснение може да се приеме, че поддържаните големи числа не могат да имат повече от 255 цифри (в съответната бройна система).

- въвеждане от стандартния вход на число с практически произволен брой цифри в:
  - десетична бройна система
  - шеснайсетична бройна система
- извеждане на екрана на голямо число в:
  - десетична бройна система
  - шеснайсетична бройна система
- събиране на две големи числа
- изваждане на две големи числа
- умножение на две големи числа
- целочислено деление на две големи числа с намиране на частно и остатък

Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на горните операции.

Въпросните операции да са реализирани в програмата чрез ясно обособени функции. Употребата на входно / изходни операции в тялото на фунцкиите е забранена (освен при функциите за вход и изход). Т.е. функциите реализират математически изображения, например `BigNumber sum (const BugNumber& a, const BigNumber& b)`.

Примери (за тестване):

"5" * "5" = "25"

"4321" - "1234" = "3087"

"1234" * "4321" = "5332114"

"12456789031415" + "98765432123456789" = "98777888912488204"

"12456789031415" * "98765432123456789" = "1230300151558439221348916026435"

"1230300151558439221348916026435" / "98765432123456789" = "12456789031415"

Геометрия

Да се напише програма, която поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на серия от операции с геометрични обекти в тримерното пространство:

- Дефиниране (въвеждане) на прави (чрез уравнения на прави) и точки (чрез координати). На правите и точките да може да се задават “имена” от една буква (f,g,h…)
- Проверка дали дадена точка лежи на дадена права
- Извежда уравнение на права по дадени две точки
- по права g и точка p извежда уравнение на права, успоредна на g и минаваща през p
- по права g и точка p, лежаща на нея, извежда уравнение на права, перпендикулярна на на g с пета в p
- по две прави намира пресечната точка, ако има такава
- по триъгълник (зададен с три точки) построява уравненията на
    - височина
    - медиана
    - симетрала

Дати

Да се напише програма, която поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на серия от операции с дати:

- въвеждане на дата от клавиатурата и проверка за коректност
- извеждане на дата на екрана в избран от потребителя формат
- по дата да се определи ден от седмицата
- по описание с ден от седмицата да се определи датата, която отговаря на него, например:
    - втората събота от месец октомври
    - последният понеделник от месец ноември
- отпечатва календар за даден месец
    - календарът да се извежда прегледно, като си личи коя дата на кой ден от седмицата отговаря

- позволява добавяне/извеждане на брой дни/седмици/месеци/години от дадена дата
    - да се обработват правилно граничните случаи, например 31 януари 2012 г. + 1 месец = 29 февруари 2012 г.
- намира разстоянието между две дати в дни/седмици/месеци/години
    - пример: разстоянието между 3 март 2012 г. и 13 юли 2013 г. е 1 година, 4 месеца, 1 седмица и 3 дена
- по дадена година определя датата на Великден за тази година

Реализацията на всички функции трябва да се съдържа в дадена програма, не е позволено използването на системна библиотека за работа с дати.

Кодирания RLE и Base64

1. RLE

Run-length encoding (RLE) представлява кодиране на низове, при което последователности от еднакви символи се заменят с двойка от символа и броя повторения. Например S = AAAABBBBCCCAABBBB може да се представи с RLE списъка R = (4,A) (4,B) (3,C) (2,A) (4,B). Да се напишат функции, които

- кодират даден низ в RLE списък
- декодират RLE списък до низ
- вмъкват един RLE списък на произволна позиция в друг, като позицията се указва в брой символи **от началото на оригиналния низ**
    - пример: след като вмъкнем (3, C) (1, B) на позиция 6 в R се получава (4,A) (2, B) (3, C) (3, B) (3, C) (2, A) (4, B)
- изтриват последователност от символи от RLE списък
    - пример: след като изтрием последователност от 8 символа от позиция 6 в R се получава списъкът (4,A) (5, B)
- проверяват дали низът, кодиран от RLE списъка A е подниз на низа, кодиран от RLE списъка B. Проверката да не включва операция по декодиране, с цел ефективност
    - пример: списъкът (2, B) (1, C) е подсписък на R
    - пример: списъкът (1,C) (3,A) (1, B) не е подсписък на R (4,B)
- По дадено RLE кодиране на низ, да се построи честотната таблица на низа (за всеки символ, който се среща в низа, се оказават броят на срещанията му в низа). Резултатът да е под формата на RLE списък, в който всеки символ се среща най-много веднъж
    - пример: (6,A) (8,B) (3,C) е честотна таблица на R

Горните операции да бъдат реализирани в програмата чрез ясно обособени функции. Употребата на входно / изходни операции в тялото на фунцкиите е забранена, освен при

функциите за вход и изход. С други думи, функциите трябва да реализират математически изображения.

2. Base64

Нека е даден масив F от байтове (unsigned char). Да се дефинират функции, които:

- Преобразуват F до низ, в който се използват само символите “A-Za-z0-9+/”.
- От кодиран по този начин низ, възстановяват съдържанието на F

Целта на кодирането е да се избегнат специалните символи като ‘\t’, ‘\n’, ‘\0’ и други. Забележете, че броят на символите в множеството “A-Za-z0-9+/” е 64 (= 26+26+10+2).

Алгоритъм на кодиране:

1) разглеждаме двоичния файл като последователност от групи по 24 бита (три байта от по 8 бита)
2) всеки 24 бита разбиваме на четири 6-битови байта
3) на всеки 6-битов байт съпоставяме съответния символ от “A-Za-z0-9+/”
4) записваме получените символи в текстовия файл

При достигане края на двоичния файл имаме от един от следните случаи:

а) при последната операция сме прочели точно 24 бита и значи всичко е ОК
б) при последното четене е имало само 8 бита
в) при последното четене е имало само 16 бита

В случай “б)” допълваме до 12 бита с нули и получаваме 2 6-битови байта, съпоставяме им символи и допълваме с два символа ‘=’ в текстовия файл.

В случай “в)” допълваме до 18 бита с нули и получаваме 3 6-битови байта, съпоставяме им символи и допълваме с един символ ‘=’ в текстовия файл.

Не е задължително да реализирате случаи “б)” и “в)”, приемете, че размера на файла е кратен на 3.

Това кодиране се нарича base64 и се използва при прикрепяне на файл към e-mail съобщение, заради изискванията на протокола да не се използват специални символи. Резултатът от кодирането е текстов файл с размер 4/3 спрямо оригиналния, но без специални символи и върши работа за всякакви файлове. Недостатък е, че съдържанието на кодирания файл е неразбираемо за човек. Съществува и друго широко използвано кодиране наречено quoted-printable – като при него се разчита, че се изпраща главно текст и малък брой специални символи, които се escape-ват.

Вашата задача е да напишете една кодираща и декодираща програма използвайки алгоритъма на base64. Програмата да реализира:

- кодиране
    - прочитане на масив от байтове (числа от 0 до 255) от клавиатурата
    - извеждане на екрана низ, който кодира масива
- декодиране
    - прочитане на низ от клавиатурата
    - извеждане на екрана на масив от байтове

Можете да проверите, че програмата ви работи правилно, като сравните резултата с този от някоя online реализация на base64.

Не се позволява използването на стандартни библиотеки и готови решения!

Подробна спецификация на base64 и quoted-printable кодиранията можете да намерите в секции 6.7 и 6.8 на rfc2045 (например на адрес http://www.faqs.org/rfcs/rfc2045.html).

Бонус: програмата да използва двоични файлове вместо масиви и да работи с командни параметри, както следва:

```
base64 encode file1.inp file1.out
base64 decode file1.out file1.copy
```

XML Parser

Да се напише програма, която разчита XML файлове и позволява правенето на прости XPath 2.0 заявки към тях.

Забележка: За проекта не е позволено използването на готови библиотеки за работа с XML. Целта на проекта е да се упражни работата със структурирани текстови файлове, а не толкова със самия XML. **Внимание:** Не се изисква осигуряване на всички условия в XML и XPath спецификациите! Достатъчно е файловете да “приличат на XML” (както файла в горния пример, който не е валиден XML), а завките да “приличат” на XPath. За уточняващи въпроси се обърнете към лекторите.

Минимални изисквания за поддържаните XPath заявки

Примерите по-долу са върху следния прост XML низ:

```
<person id=”0”>
     <name>John Smith</name>
     <address>USA</address>
</person>
<person id=”1”>
```

```
    <name>Ivan Petrov</name>
    <address>Bulgaria</address>
</person>
```

- да поддържат оператора / (например “person/address” дава списък с всички адреси във файла)
- да поддържат оператора [] (например “person/address[0]” два адресът на първия елемент във файла)
- да поддържат оператора @ (например “person(@id)” дава списък с id на всички елементи във файла)
- Оператори за сравнение = (например “person(address=”USA”)/name” дава списък с имената на всички елементи, чиито адреси са “USA”)

Силоз 3: Информационни системи

Библиотека

Да се напише компютърна програма, реализираща информационна система, която поддържа библиотека. Програмата съхранява и обработва данни за наличните в момента книги под формата на текстови файлове.

Всяка книга се характеризира със следните данни:

- автор
- заглавие
- жанр
- кратко описание
- година на издаване
- ключови думи
- рейтинг
- уникален номер за библиотеката

Системата поддържа два вида потребители — администратори и клиенти на библиотеката. Всеки потребител се характеризира със следните данни:

- потребителско име
- парола
- ниво на достъп — указва дали потребителят е администратор или не.

Програмата работи в интерактивен текстов режим, като очаква въвеждането на команди от потребителя и реагира на тях по съответен начин. Всички команди са предварително дефинирани.

Команди достъпни от всички потребители:

login
След въвеждането на командата потребителят последователно е питан за потребителско име и парола. Ако потребител с посочените данни съществува в програмата, се извежда съобщение “Welcome, <username>!”, където <username> съответства на потребителското име. В противен случай се извежда съобщение за грешно име или парола. При повторен опит за login, се изкарва съобщение “You are already logged in.”

logout
потребителят напуска системата (програмата продължава да работи)

help
извежда на екрана списък с всички възможни команди и краткото им описание

exit
програмата спира изпълнение (Press any key to continue . . . :)

books all
извежда последователно за всяка книга следната информация:
- заглавие, автор, жанр, персонален номер

books info <isbn_value>
извежда на екрана подробна информация за книга с персонален номер равен на <isbn_value>
Пример: books info 1124

books find <option> <option_string>
<option**>** е един от следните низовете {“title” , “author”, “tag”}
<option_string**>** е стойноста на критерия за търсене, може да съдържа интервали
Пример: books find title Introduction to programming
books find Stephen King
books find superhero

books sort <option> <option_string>
<option> е един от следните стрингове {“year”, “rating”}
Извежда информация за книгите на екрана, в сортиран вид спрямо стойността на option.

При първоначално стартиране на програмата няма налични данни за книги. Има регистриран по подразбиране само един потребител с администраторски акаунт със следните данни:

- потребителско име: “admin”

- парола: “i<3c++”

Програмата очаква да се въведе команда, като след въвеждането и се изпълнява според дефинираните правила. Това продължава до въвеждането на командата “exit”, която прекратява програмата.

команда	логин	админ
help	не	не
login	не	не
logout	да	не
exit	не	не
books all	да	не
books find	да	не
books sort	да	не
books view	да	не
books add	да	да
books remove	да	да
users add	да	да
users remove	да	да

Бонус:

- при въвеждане на паролата на екрана да се изписва символа * вместо реалния символ
- при сортиране на книгите по зададен критерий, да се напише алгоритъм различен от пряка селекция и метода на мехурчето
- Търсене на книга по зададен критерий да игнорира регистъра на буквите (малки или големи)

Хотел

Да се напише компютърна програма, реализираща информационна система, обслужваща

хотел. Програмата съхранява и обработва необходимите данни под формата на текстови файлове. Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на следните операции:

- регистриране в стая (задава се номер на стая, начална и крайна дата и се въвежда коментар, например “семейство Симпсън”)
- списък на свободни стаи
- освобождаване на заета стая
- справка за използването на стаи в даден период (по начална и крайна дата се извежда списък, в който за всяка стая, използвана в дадения период, се извежда и броя на дните, в които е била използвана)
- намиране на подходяща (с необходимия брой легла) свободна стая по дадена начална и крайна дата
- задаване на допълнителни изисквания за стаи (бебешка кошарка, със/без закуска, с изглед към морето/планината/двора)
- да се напише алгоритъм, който предлага спешно намиране на стая за важен гост в случай на липса на свободни стаи за даден период. Алгоритъмът да предлага разместване на настанените от най-много две стаи

Склад

Да се напише компютърна програма, реализираща информационна система, обслужваща склад. Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на следните операции:

- списък на наличните продукти в склада. За всеки продукт се съхранява и извежда следната информация:
    - име (описание - символен низ с произволна дължина)
    - срок на годност
    - дата на постъпване в склада
    - има не производител
    - мерна единица (килограми, литри)
    - налично количество
    - местоположение (секция/рафт/номер)
    - коментар (свободен текст)
- записване на нова доставка (по име и количество се обновява наличното количество в склада)
- изваждане на продукти от склада (по дадено име и изтеглено количество се намалява съответното количество в склада)
    - да се извежда грешка при опит за изваждане на неналични продукти
- справка за наличността в даден период (по дадена начална и крайна дата се

извежда списък с всички промени на наличността в дадения период, включително зареждания и извеждания на стоки)
- разчиства склада от всички стоки, на които е изтекъл или предстои скоро да изтече срока на годност, като извежда информация за разчистените стоки

Бонус: Програмата да съхранява и обработва необходимите данни под формата на текстови файлове.

СУСИ

Да се напише компютърна програма, реализираща информационна система за обслужване на студенти. Програмата съхранява и обработва необходимите данни под формата на текстови файлове. Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на следните операции:

- записване на студент към системата, чрез съхранение на следната информация:
  - Име — символен низ с произволна дължина
  - Факлутетен номер
  - Година на приемане
  - Текущо записан курс, специалност, група
  - всякаква допълнителна информация, нужна за изпълнение на заявките по-долу
- промяна на статус (прекъсване на обучението, завършване)
- справка за студенти в даден поток/курс/специалност (по дадени поток, курс и специалност се извеждат всички съответни студенти)
- добавяне на оценки/изпити (за даден предмет, оценка и студент)
- записване на задължителни/избираеми предмети
- отпечатване на протоколи
- академична справка за оценките на даден студент (списък с всички взети изпити и съответните оценки). Да се включи също и списък с невзетите изпити (предмети, които са записани, но за тях няма оценки), ако има такива

Бонус: Програмата да съхранява и обработва необходимите данни под формата на текстови файлове.

Билети

Да се напише компютърна програма, реализираща информационна система, която обслужва билетна каса. Програмата съхранява и обработва необходимите данни под формата на текстови файлове. Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на следните операции:

- добавяне на представление (име и дата)
- списък със свободни места за дадено представление (непродадени и незапазени билети)
- запазване на билет (въвежда се номер на място и коментар, примерно “За Хоумър Симпсън”)
- закупуване на билет. За всеки билет се издава уникален сложен код, който съдържа информация за съответното място
- получаване на списък на запазените, но неплатени (незакупени) билети (за дадено представление или за всички представления)
- отмяна на резервация
- проверка за валидност на билет (по даден код се извлича номера на мястото или се връща грешка, ако кодът е невалиден)
- справка за закупени билети в даден период (по дадена начална и крайна дата се извежда списък с всички изнесени представления като за всяко отделно представление се извежда и количеството продадени билети)

Бонус: Програмата да съхранява и обработва необходимите данни под формата на текстови файлове.

Личен календар

Да се напише компютърна програма, реализираща информационна система, която поддържа личен календар. Програмата да поддържа текстов диалогов режим, позволяващ удобен интерактивен избор на следните операции:

- запазване/отказване на час за среща. Записва се начало на среща (ден, час), край на среща, име (кратко описание - например “зъболекар”) и коментар
- отпечатване на дневна програма, като по даден ден се извежда хронологичен списък с всички ангажименти за деня
- търсене на среща: по име се извеждат останалите данни се срещата
- определяне на някои дни като неработни (въвежда се коментар, напр. “ден на труда”)
- извеждане на статистика за натовареност: по дадени начална и крайна дата се извеждат подреден списък с дните от седмицата по критерия “брой заети часове”
- намиране на свободно място за среща: по дадена начална дата и желана продължителност на срещата търси дата, на която е възможно да се запази такава среща, но само в работни дни и не преди 8 часа сутринта или след 5 часа вечерта.

Бонус: Програмата да съхранява и обработва необходимите данни под формата на текстови файлове.