7 października 2010

Kurs języka Ruby

Lista 1.

Zadanie 1. Zaprogramuj

1. trójargumentową funkcję, gdzie pierwszy argument to rok, drugi: miesiąc, trzeci: dzień miesiąca (podawane jako liczby całkowite), która oblicza ile zostało dni do Sylwestra 2010 od daty podanej jako argument;

2. funkcję, której argumentem jest lista list liczb całkowitych, a wynikiem suma wszystkich liczb. Nie jest wymagana kontrola poprawności typów;

3. funkcję, której argumentami są dwie trójelementowe listy liczb całkowitych. W listach pamiętane są daty w postaci *[dd, mm, rrrr]*. Wynikiem jest liczba dni dzielących te daty. Możesz przyjąć, że argumenty są poprawnymi datami;

4. funkcję, której argumentem jest liczba, a wynikiem słowny zapis liczby. Możesz przyjąć, że wynik jest uproszczony, np. wynikiem $f(123)$ jest *jeden dwa trzy*;

5. funkcję, której argumentem jest lista słów (stringów), gdzie słowa są z ustalonego zbioru słów { *zero, jeden, dwa, trzy, cztery, ..., dziewięć* }. Wynikiem ma być liczba, np. *f(["trzy", "pięć", "siedem"])* ma dać 357;

6. funkcję wypisującą na konsoli *trójkąt Pascala*. Argumentem funkcji jest liczba wierszy trójkąta do wypisania;

7. funkcję, której argumentami są dwie liczby całkowite oznaczające godzinę i minutę. Wyikiem działania funkcji ma być string zwracający tę godzinę w potocznej formie. Przykładowo `godzina(12,45)` powinna zwrócić *"za kwadrans pierwsza"*.

Najwygodniej jest umieścić wszystkie rozwiązania w jednym pliku, a na końcu umieścić kilka testów prezentujących możliwości zaimplementowanych funkcji. Za każde zadań można otrzymać 0,5 punktu, jednak za całą listę nie można otrzymać więcej niż 2 punkty (a więc do oceny można przedstawić co najwyżej 4 funkcje). Termin: zajęcia w przyszłym tygodniu.

Marcin Młotkowski

14 października 2010

Kurs języka Ruby

Lista 2.

Zadanie 1. Zaimplementuj jakąś reprezentację grafu nieskierowanego za pomocą słowników. Przyjmij, że wierzchołki są etykietowane elementami typu `String`, oraz że różne wierzchołki mają różne etykiety. Napisz procedurę `sciezka(graf, a,b)`, której argumentami są: graf i etykiety, a wynikiem lista dróg (tablica tablic etykiet wierzchołków) z a do b lub lista pusta jeśli droga nie istnieje.

Jeśli w grafie jest cykl, to można przyjąć następujące ograniczenia (do wyboru):

- żadne dwie ścieżki w rozwiązaniu nie zawierają wspólnych krawędzi;
- żadna ścieżka nie zawiera dwa razy tej samej krawędzi.

Zadanie 2. Napisz program do dekodowania napisów zapisanych alfabetem Morse'a. Wymagane jest, aby alfabet był pamiętany za pomocą drzewa binarnego. Najprostsze rozwiązanie za pomocą tablic haszujących nie będzie sprawdzane.

Zadanie 3. Pewien prosty system zapisów na zajęcia zapamiętuje dwa rodzaje danych: listę osób zapisanych na każde zajęcia (oraz godziny tych zajęć), oraz listę zajęć prowadzonych w danej sali. Jednak konieczne jest sprawdzenie, czy informacje są pełne i niesprzeczne. Napisz trzy funkcje:

1. sprawdzającą, czy przypadkiem ktoś nie zapisał się na zajęcia odbywające się w tym samym czasie;
2. sprawdzającą, czy wszystkie zajęcia są przypisane do jakiejś sali;
3. sprawdzającą, czy zajęcia przypisane salom się nie nakładają.

Oczywiście, wcześniej trzeba zaproponować struktury danych przechowujące te dane.

Zadanie 4. Wiele osób twierdzi, że swoje sukcesy osiągnęły dzięki planowaniu swoich zajęć. Zajęcia można podzielić na dwie podstawowe kategorie: zajęcia o ustalonych godzinach rozpoczęcia i zakończenia (tak jak wykład z Ruby-ego ;), oraz zajęcia, na które trzeba przeznaczyć określoną ilość czasu.

Zaprojektuj strukturę danych przechowującą informacje o przewidywanych zajęciach. Napisz funkcję, która ułoży plan zajęć na najbliższy czas dbając, aby nie zaczynać zbyt wcześnie pracy i aby nie pracować dłużej niż do jakiejś godziny. Jeżeli będą wolne luki w planie, niech ta funkcja zaproponuje jakieś miłe i relaksujące zajęcia.

Każde zadanie jest warte 2 punkty, na zajęciach oddaje się jedno zadanie.

Marcin Młotkowski

21 października 2010

Kurs języka Ruby

Lista 3.

Zadanie 1. (2 pkt) Bloki z jednym parametrem można traktować jak definicje jednoargumentowych funkcji. Korzystając z tej obserwacji zaprogramuj dwie procedury. Pierwsza z nich `calka(a,b,&b)` powinna obliczać numerycznie całkę oznaczoną na przedziale $[a, b]$ funkcji zadanej jako blok. Dokładność obliczeń może być ustalona. Druga funkcja to `wykres(a, b, &blok)`, która za pomocą znaków ASCII naszkicuje wykres funkcji danej jako blok. Można przyjąć arbitralny rozmiar terminala.

Implementacje poniższych funkcji powinne być w postaci jednego wyrażenia. Jest to możliwe używając tylko zakresów, operacji na tablicach i bloków. W przypadku bardzo długich wyrażeń akceptowane będzie podzielenie rozwiązania na podwyrażenia.

Zadanie 2. (1 pkt) Napisz jednoargumentową funkcję `pierwsza(n)`, która zwraca tablicę liczb pierwszych nie większych niż n.

Zadanie 3. (1 pkt) Napisz jednoargumentową funkcję `doskonale(n)`, która zwraca tablicę liczb doskonałych nie większych niż n, na przykład

```
doskonale(1000)
==> [6, 28, 496, 8128]
```

Zadanie 4. (1 pkt) Napisz jednoargumentową funkcję `rozklad(n)` która oblicza rozkład liczby n na czynniki pierwsze i zwraca jako wynik tablicę tablic $[[p_1, w_1], [p_2, w_2], \ldots, [p_k, w_k]]$ taką, że
$n = p_1^{w_1} * p_2^{w_2} * \ldots * p_k^{w_k}$ oraz $p_1, \ldots, p_k$ są różnymi liczbami pierwszymi. Na przykład

```
rozklad(756)
==> [[2, 2], [3, 3], [7, 1]]
```

Zadanie 5. (1 pkt) Napisz jednoargumentową funkcję `zaprzyjaznione(n)`, która zwraca tablicę par liczb zaprzyjaźnionych nie większych niż n, na przykład

```
zaprzyjaznione(1300)
==> [[220, 284], [1184, 1210]]
```

Dodatkowe wyjaśnienia można znaleźć w polskiej Wikipedii.

Za rozwiązanie powyższych zadań można uzyskać co najwyżej 2 pkt.

Marcin Młotkowski

28 października 2010

Kurs języka Ruby

Lista 4.

Zadanie 1. Typowym zadaniem na zajęciach z informatyki jest oprogramowanie wypożyczalni (płyt, samochodów etc) czy biblioteki. Zazwyczaj funkcjonalności czy schematy takich programów są bardzo podobne. Zdefiniuj więc moduł wypożyczeń implementujący: instytucję wypożyczającą, przedmioty do wypożyczenia oraz osoby korzystające z usług tejże instytucji. Zaprogramuj funkcje: wypożyczenia, oddania i wyszukiwania przedmiotów, które jednocześnie wysyłają komunikaty na konsolę o wykonaniu (lub odmowie wykonania) operacji. Utwórz za pomocą mechanizmu *mix-in* bibliotekę.

Zadbaj o personalizację komunikatów, tj. aby komunikaty były postaci „wypożyczono książkę ...” a nie „wypożyczono obiekt #<Ksiazka:0x7fd04e280f20>”, o ile oczywiście przykładową klasą będzie biblioteka.

Zadanie 2. Ważnym choć czasem niedocenianym elementem rozwijania oprogramowania jest testowanie. Obiekty można na przykład testować dodając odpowiednie metody testujące. Dla wygody można przyjąć, że nazwy metod testujących zaczynają się od `test_`.

Rozszerz podany na wykładzie mix-in ***Debug*** o procedurę `check`, która wyszukuje w klasie wszystkie metody zaczynające się na `test_` i wykonująca je. Przyjmij, że metody `test_*` zwracają wynik testu w postaci stringa.

Zadanie 3. Zaprogramuj klasę ***DrzewoBinarne*** wraz z operacjami `wstaw`, `istnieje?` i `usun`. Przyjmij, że elementy drzewa są obiektami klasy ***Element***. Podaj przykład wykorzystania tych klas.

Każde zadanie jest warte 2 punkty. Na pracowni oddaje się jedno zadanie.

Marcin Młotkowski

4 listopada 2010

Kurs języka Ruby

Lista 5.

Zadanie 1. Zaprogramuj pakiet funkcji przeglądających istniejące serwisy WWW. Pakiet powinien zawierać ogólną funkcję przeglądania stron `przeglad(start_page, depth, block)`, gdzie `start_page` to adres strony startowej, `depth` to głębokość z jaką należy przeglądać serwis, a `block` to jednoargumentowy blok o argumencie typu ***String***. Funkcja ta powinna przeglądać strony serwisu od podanej strony, i dla każdej z nich wykonać instrukcje zawarte w bloku. Zaprogramuj dwie funkcje:

- `page_weight(page)`, która oblicza liczbę elementów wpływających na czas ściągania i renderowania strony, takich jak obrazki czy aplety;
- `page_summary(page)`, która dla każdej strony wypisuje informację o stronie uzyskaną z nagłówka `<head>`, tj. tytuł, opis, autor, słowa kluczowe etc.

Zadanie 2. Napisz własną wyszukiwarkę, która przegląda wybrany serwis internetowy i zapamiętuje wystąpienia poszczególnych słów, oraz umożliwia wyszukiwanie słow zadanych jako wyrażenie. Kod wyszukiwarki powinien mieć postać modułu zawierającego funkcje:

- `index(start_page, depth)` która przegląda od podanej strony oraz indeksuje słowa ze strony;
- `search(reg_exp)`, która podaje listę stron na których występują słowa pasujące do `reg_exp`, oczywiście korzystając wyłącznie z zebranej wcześniej informacji.

Zadanie 3. Zaprogramuj funkcję `distance(page_a, page_b)`, która oblicza odległość od strony o url'u `page_a` do strony o url'u `page_b`, przy czym odległość jest rozumiana jako liczba kliknięć w kolejne odnośniki. Oczywiście można przyjąć jakieś arbitralne ograniczenie na czas przeszukiwania.

Zadanie 4. Zaprogramuj pakiet służący do monitorowania wybranych stron i informujący o zmianie treści strony. Na początku podaje się listę stron jakie mają być monitorowane, następnie program oblicza sumę kontrolną np. md5. Następnie co jakiś czas wskazane strony są ponownie odczytywane, i jeżeli nastąpiła zmiana wysyłany jest komunikat na konsolę. Uzupełnij program o możliwość zapisywania stanu sesji (tj. stron oraz sum kontrolnych) i jej odtwarzania. Do tego przydatny będzie moduł YAML.

Choć treść powyższe zadań sugeruje rozwiązanie w postaci modułu, proszę zaprogramowane funkcje "opakować" w klasy i obiekty. Każde z tych zadań jest warte 3 punkty, na zajęciach proszę oddać jedno zadanie.

Marcin Młotkowski

18 listopada 2010

Kurs języka Ruby

Lista 6.

Do poniższych zadań proszę dołączyć interaktywny miniinterfejs do obsługi poniższych programów; wystarczy jeśli się skorzysta z `gets` i `puts`. Poniższe zadania powinne implementować przeglądanie, wyszukiwanie, dodawanie i usuwanie wpisów.

Zadanie 1. Zaprogramuj własny organizator swojego czasu zawierający planowane spotkania (od-do), sprawy do załatwienia (do czasu), wraz z opcją przypominania. Dane niech będą przechowywane na dysku, np. korzystając z dbm'a, SQL'a czy YAML'a.

Zadanie 2. Napisz program, który przechowuje w swojej lokalnej bazie danych informacje o posiadanych płytach z muzyką (identyfikator płyty, lista utworów i autorzy) wraz z informacjami o wypożyczeniu płyty znajomym.

Zadanie 3. Zaprogramuj własny notatnik z kontaktami do znajomych zawierający ich numery telefonów, adresy email czy gg. Dane niech będą przechowywane w bazie danych (typu dbm, pickle czy SQLite).

Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 3 punkty. Proszę zadbać też o komentarze w swoich programach.

Marcin Młotkowski

25 listopada 2010

Kurs języka Ruby

Lista 7.

Zadanie 1. Zaprogramuj serwer logów (drb), który będzie przechowywał nadesłane komunikaty w bazie danych wraz z czasem ich otrzymania. Serwer powinien implementować funkcję `save(prg_id, msg)`, gdzie `prg_id` jest identyfikatorem programu (serwer może zbierać logi z różnych aplikacji), a `msg` oznacza obiekt klasy ***String***. Dodatkowo zaimplementuj na serwerze metodę `raport(od, do, prg_id, re)`, gdzie `od` i `do` są obiektami klasy ***Time*** definiującymi zakres wyszukiwania, `prg_id` jest identyfikatorem programu, a `re` wyrażeniem regularnym. Funkcja powinna zwracać obiekt ***String***, będący html'em. Sposób przechowywania (dbm, sqlite3 etc.) jest dowolny.

Zadanie 2. Zaimplementuj repozytorium do przechowywania obiektów, tj. serwer implementujący metody `store(obj, id)`, `restore(id)` i `delete(id)`, gdzie `id` jest dowolnym identyfikatorem obiektu.

Zaprogramuj również metody: `stan`, która zwraca html (jako string) z informacją o zapisanych obiektach, tj. klasę obiektu wraz ze stanem jego pól, oraz wyszukiwarkę obiektów implementujących daną jako argument listę metod.

Zadanie 3. Serwer drb może być wykorzystywany jako narzędzie do zdalnego monitorowania i zarządzania komputerami, tj. na każdym komputerze klienckim jest uruchomiony serwer z metodami, które wywołują polecenia systemowe sprawdzające podstawowe wartości, takie jak sprawdzenie obciążenia procesora czy ilość wolnego miejsca na dysku. Zaprogramuj taki serwer wraz z klientem, który mając daną listę komputerów będzie je odpytywał co jakiś czas i raportował ich stan. Zbadaj, jaki musi być ustawiony poziom zmiennej `$SAFE`.

Na zajęcia należy wykonać jedno z tych zadań. Każde zadanie jest warte 3 punkty. Proszę zadbać też o komentarze w swoich programach.

Marcin Młotkowski

2 grudnia 2010

Kurs języka Ruby

Lista 8.

Zadanie polega na uzupełnieniu wcześniej zaprogramowanych zadań o wątki i testy jednostkowe. Dobrym kandydatem są np. zadania z listy 5. W przypadku testów można wybrać inne zadanie niż wybrane dla wątków. Wymagane jest stworzenie przynajmniej 3 metod z testami.
Za to zadanie można otrzymać do 3 punktów.

Marcin Młotkowski

9 grudnia 2010

Kurs języka Ruby

Lista 9.

Zadanie 1. Zaprogramuj interfejs graficzny (np. **tk**) do jednego z zadań z listy 6.

Zadanie 2. Zaprogramuj program rysujący wykresy kilku ustalonych funkcji. Przyjmij, że wskazanie funkcji następuje w menu, natomiast wartości *od do* przedziału rysowania są określane w kontrolkach ***Entry***.

Zadanie 3. Zaprogramuj następującą prostą grę: na rysunku jest armata, która ma regulowany kąt wystrzału i prędkość początkową pocisku, oraz cel (odległość między armatą i celem może być losowa). Kąt wystrzału oraz prędkość pocisku powinna być zadawana przez użytkownika, np. za pomocą kontrolek ***Entry***. Zadanie polega na takim wybraniu kąta i prędkości, aby pocisk trafił w cel. Korzystając z prostych praw fizyki narysuj tor pocisku oraz oblicz, czy pocisk trafił w cel.

Zadanie 4. Bardzo ładnymi figurami geometrycznymi są fraktale. Sporo materiałów o nich można znaleźć w internecie (są również książki o fraktalach w naszej bibliotece). Zadanie polega na zaprogramowaniu kilku fraktali. Fraktale zwykle mają parametry, które powinny być podawane np. poprzez kontrolki ***Entry***.

Każde zadanie jest warte 3 pkt., jak zwykle można oddać tylko jedno.

To już jest ostatnia lista zadań. Proszę zastanowić się nad tematem projektu i uzgodnić go z prowadzącym pracownię.

Marcin Młotkowski