Języki i paradygmaty programowania: Laboratorium nr 9

Podstawowe paradygmaty programowania obiektowego - wprowadzenie. Wyrażenia regularne.

2016-2017

mgr inż. Przemysław Walkowiak
dr inż. Michał Ciesielczyk

Instrukcja

W czasie pisania programu pamiętaj o:

1. dbaniu o czytelność kodu (odpowiednie formatowanie kodu, nazewnictwo zmiennych adekwatne do ich znaczenia, komentarze),
2. dbaniu o czytelność interfejsu z użytkownikiem (w sposób jawny pytaj użytkownika jakie dane ma podać oraz opisuj wyniki, które zwracasz),
3. przed fragmentem implementującym poszczególne zadania umieść komentarz: `/*Zadanie X */` oraz wypisz na ekranie analogiczny komunikat (X jest numerem zadania): `std::cout << "Zadanie X"<< std::endl;`,
4. każde zadanie umieść w oddzielnej funkcji (w niej dopiero należy odwoływać się do zaimplementowanych funkcji i klas),
5. zaimplementuj menu wyboru zadania, a następnie wykorzystując pętle **do–while** oraz konstrukcję **switch** wykonaj odpowiedni fragment kodu,
6. w zadaniach wymagających udzielenia komentarza bądź odpowiedzi, należy umieścić go w kodzie programu (np. w postaci komentarza albo wydrukować na ekranie),
7. w zadaniach polegających na zaprojektowaniu klasy należy utworzyć jej instancję i wykorzystać zaimplementowaną funkcjonalność.

Wprowadzenie

Zadania

Zadanie 1

Ściągnij plik `wyrazenia_regularne.cpp`, a następnie przeanalizuj funkcję `zadanie1`. Zdefiniowane wyrażenie regularne `speed_regex` jest dopasowywane do danych ze zmiennej `phys_data` (ona jest z kolei zdefiniowana na początku pliku źródłowego).

```
Speed: 366
Mass: 35
Point 50 70
Speed: 378
Mass: 32
Point 25 35
Speed: 400
Mass: 30
```

```
Point 23 16
```

Napisz analogiczny kod wraz z odpowiednim wyrażeniem regularnym, który dopasuje obie wartości liczbowe z wiersza `Point` i je wyświetli na ekranie.

Zadanie 2

Przeanalizuj funkcję `zadanie2`. Utwórz strukturę `Point` z dwoma polami `x,y`. Zaimplementuj funkcję `std::vector<Point> parse_points(std::string data)`, która znajdzie wszystkie wystąpienia wierszy z prefiksem `Point` i zwróci kolekcję z elementami typu `Point`. Zaimplementuj funkcję `void print(const std::vector<Point>& points)` wyświetlającą kolekcję na ekranie.

Zadanie 3

Przeanalizuj funkcję `zadanie3`. Operuje ona na danych z pliku `cats.html` (fragment strony `http://www.cutestpaw.com/articles/50-cute-cats-make-your-life-happier/`). Otwórz ten plik w edytorze tekstu (np. notatnik lub notepad++) i przyjrzyj się jego zawartości.

Funkcje `parse_titles` i `parse_titles_it`, dwoma sposobami wyciągają ze znaczników HTMLowych `<img>` i `<a>` wartość atrybutu `title`. Dostosuj wyrażenia regularne tak, aby wyciągnąć tytuł tylko ze znacznika `<img>`.

Zadanie 4

W pliku źródłowym została zdefinowana struktura `Cat` wraz z kilkoma funkcjami pomocniczymi. Zaimplementuj funkcję `std::vector<Cat> parse_cats(std::string cats_data)`, wyciągającą z kodu HTML dla każdego zdjęcia następujące informacje:

- zawartość atrybutu `href` znacznika `<a>` – wpisz do pola `url` struktury `Cat`,
- zawartość atrybutu `src` znacznika `<img>` – wpisz do pola `img_url` struktury `Cat`,
- zawartość atrybutu `title` znacznika `<img>` – wpisz do pola `title` struktury `Cat`.