24 lutego 2021

Programowanie obiektowe

Lista 1.

Zaprogramuj dwa z poniższych zadań w języku w C, Pascalu czy w Pythonie, jednak bez deklarowania własnych klas i nie wykorzystując klas dostępnych w C++ czy w Pythonie. W przypadku Pythona, gdzie jedynym sposobem definiowania typu jest deklaracja klasy, można poniższe zadania zaimplementować wykorzystując krotki bądź listy, które będą przekazywane do funkcji jako argumenty.
Wyliczenia w Pythonie można uzyskać np. tak:

```
class Enum:
    Zero, One, Two = range(3)

var_name = Enum.One
```

Za każde zadanie na tej liście można otrzymać do 3 punktów.

Zadanie 1

Zadeklaruj typ `typedef Figura ...` który będzie reprezentował figury geometryczne: trójkąt, koło lub kwadrat [1], wraz z ich położeniem w dwuwymiarowym układzie współrzędnych.
Przyjmij, że pole `typfig` typu wyliczeniowego wyznacza rodzaj reprezentowanej figury geometrycznej. Wygodnie jest mieć funkcje inicjujące takie figury, wtedy utworzenie odpowiedniej figury wygląda po prostu tak:

```
Figura *f;
f = new_square(1.0, -1.0, 3.0);
```

Zadbaj o kontrolę danych, np. próba utworzenia koła o ujemnym promieniu powinna poinformować o problemie.
Zdefiniuj również trzy funkcje:

- `float pole(Figura *f)`: wylicza pole figury `f`;
- `void przesun(Figura *f, float x, float y)`: przesuwa `f` o wektor (x, y);
- `float sumapol(Figura* f[], int size)` oblicza sumę pól figur znajdujących się w tablicy (`size` jest rozmiarem tablicy).

Na przykład wywołanie funkcji

```
pole(f)
```

powinno zwrócić 9 dla `f` z przykładu powyżej. Podczas oceny programu wskaż miejsca, które wymagają modyfikacji, jeśli rozszerzymy typ Figura o trapezy.

Zadanie 2

Zaprojektuj własny typ `Ulamki` reprezentujące ułamki, gdzie licznik i mianownik są liczbami całkowitymi, tj. zaprogramuj:

- odpowiednie struktury danych służące do przechowywania wartości tego typu;
- funkcję `Ulamki * nowy_ulamek(float num, float denom)` tworzącą nowy ułamek w postaci *uproszczonej*;

[1] można przyjąć, że boki kwadratu czy jeden z boków trójkąta są równoległe do osi układu współrzędnych

- cztery podstawowe operacje arytmetyczne na ułamkach. Zaimplementuj dwie możliwe realizacje takich funkcji: w pierwszej funkcja zwraca wskaźnik do nowoutworzonego elementu tego typu; w drugiej funkcja modyfikuje drugi z argumentów.

Zadanie 3

Zaprogramuj własną tablicę indeksowaną dowolną liczbą całkowitą wraz z operacjami wstawiania i pobierania elementów z podanej pozycji w tablicy. Po utworzeniu takiej tablicy jest ona długości zero. Po wykonaniu pierwszej operacji wstawienia na jakąś pozycję (powiedzmy, 2), tablica jest już jednoelementowa. Kolejne wstawienie, na przykład na pozycję -3, powoduje rozszerzenie tablicy tak, że zawiera ona pozycje o indeksach [-3..2]. Na przykład jeśli wykonamy poniższy kod

```
Tablica *t;
t = nowa_tablica();
dodaj(t, 2, 6.0);
dodaj(t, -3, 5.0)
```

to możemy odwołoać się do elementu o indeksie `-1`:

```
indeks(t, -1)
```

Oczywiście, zwrócona wartość może być albo domyślna (zero lub coś podobnego), albo losowa. Przyjmij, że w tablicy przechowujemy wartości dowolnego typu zdefiniowanego za pomocą deklaracji typedef. Przy okazji, wkładając wartości do tej tablicy wkładamy wartość czy jej kopię?

Zadanie 4

Zaprogramuj funkcje do kodowania i dekodowania komunikatów zapisanych alfabetem Morse'a. Wymagane jest wykorzystanie jakiejś struktury danych, która umożliwi wyszukiwanie odpowiednich ciągów szybciej niż w czasie liniowym Można się ograniczyć do liter alfabetu angielskiego[2] i cyfr.

Dodatkowe wymaganie do wszystkich powyższych zadań: powyższe zadania powinne być zaimplementowane w postaci modułu bądź biblioteki. Częścią rozwiązania powinien być krótki przykład (w odrębnym pliku) korzystający z zaimplementowanego modułu.

Oceniane i punktowane są jedynie dwa zadania.

Marcin Młotkowski

[2]czasem używa się określenia *alfabet łaciński*, który w czasach starożytnych nie zawierał np. liter W czy U