23 lutego 2024

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. Oceniane i punktowane są jedynie dwa zadania.
Uwaga: na stronie wykładu w systemie SKOS znajdują się *Zalecenia dot. wysyłanych zadań.* Proszę się z nimi zapoznać przed wysłaniem swoich programów.

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 cztery funkcje:

- `float pole(Figura *f)`: wylicza pole figury `f`;
- `void przesun(Figura *f, float x, float y)`: przesuwa `f` o wektor (x, y);
- `void show(Figura *f)`: wypisuje informacje o figurze (typ, położenie);
- `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 `Ulamek` reprezentujące ułamki, gdzie licznik i mianownik są liczbami całkowitymi, tj. zaprogramuj:

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

- odpowiednie struktury danych służące do przechowywania wartości tego typu;
- funkcję `Ulamek * nowy_ulamek(int num, int denom)` tworzącą nowy ułamek w postaci *uproszczonej*;
- funkcję `void show(Ulamek *u)` wypisującą wartość ułamka;
- 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`.

Zaprogramuj pomocniczą funkcję `void show(Tablica *t)` wypisującą listę.

Zadanie 4

Zaprogramuj funkcję

`void tabliczka(float x1, float x2, float y1, float y2, float skok)`

drukującą na ekranie *tabliczkę mnożenia* dla zadanego przedziału $[x1, \ldots, x2) \times [y1, \ldots, y2)$, ale dla liczb typu *float* z podanym skokiem. Przykładowo, wywołanie

```
tabliczka(0.2, 1.3, 0.2, 3.14, 0.3);
```

powinno dać wynik

```
       0.20 0.50 0.80 1.10 1.40 1.70 2.00 2.30 2.60 2.90
0.20:  0.04 0.10 0.16 0.22 0.28 0.34 0.40 0.46 0.52 0.58
0.50:  0.10 0.25 0.40 0.55 0.70 0.85 1.00 1.15 1.30 1.45
0.80:  0.16 0.40 0.64 0.88 1.12 1.36 1.60 1.84 2.08 2.32
1.10:  0.22 0.55 0.88 1.21 1.54 1.87 2.20 2.53 2.86 3.19
```

W zadaniu ważne jest wypisanie nagłówków wierszy i kolumn oraz odpowiednie sformatowanie liczb tak, aby kolumny były wyrównane. Można tu przyjąć, że argumentami są tylko liczby nieujemne.

Marcin Młotkowski