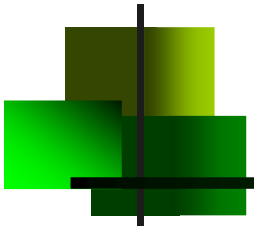
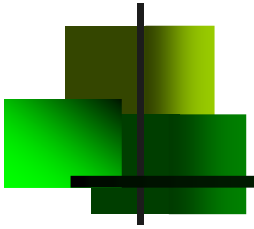


Metody programowania - ćwiczenia

Polecenia

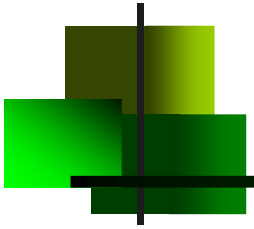


Ćwiczenia 0 – funkcje i tablice - powtórzenie

1. Napisać program, w którym funkcja main() wywołuje funkcję (zdefiniowaną przez programistę), która przyjmuje argument w skali Celsjusza i zwraca jej odpowiednik w skali Fahrenheita. Program powinien prosić użytkownika o wpisanie wartości w skali Celsjusza. (Wskazówka: stopnie Fahrenheita = 1.8 * stopnie Celsjusza + 32.0)

2. Napisać program realizujący obsługę danych dotyczących darowizn, zapisanych w tablicy. Program powinien:
   - wczytywać (z klawiatury) do tablicy typu double wartości co najwyżej 100 darowizn,
   - wyświetlać średnią wartość darowizn
   - informować, wartości których darowizn są większe od średniej.

   W programie powinny zostać zdefiniowane przynajmniej 3 funkcje, każda powinna przyjmować jako parametr tablicę darowizn. Funcja obliczająca średnią powinna zwracać wartość obliczonej średniej. Funkcja wypisująca darowizny o wartości większej od średniej powinna przyjmować średnią jako parametr.



1. Rekurencja

Napisać i uruchomić program wykonujący obliczenia za pomocą poniższych funkcji rekurencyjnych; wyświetlić wyniki otrzymane przez poszczególne funkcje oraz informację o liczbie wywołań tych funkcji:

a. **silnia(int n) – funkcja obliczająca wartość silni liczby n; przeprowadzić obliczenia dla dwóch wartości n (np. n=7, n=16);**
b. **nwd(int a, int b) – funkcja obliczająca największy wspólny dzielnik liczb a i b; przeprowadzić obliczenia dla czterech par wartości a i b;**
c. **fibonacci(int n) – funkcja obliczająca wartość n-tej liczby Fibonacciego; przeprowadzić obliczenia dla dwóch wartości n (uwzględniających przypadki, gdy n<20 i n >40, np. n=17, n=45);**
d. **hanoi(int n,char a,char b, char c) – funkcja realizująca układanie wież hanoi; przeprowadzić obliczenia dla dwóch wartości n (uwzględniających przypadki, gdy n<10 i n >20).**

Wyniki przeprowadzonych obliczeń należy zapisać w zeszycie lub epliku.

2. Wskaźniki (1/2)

1. Napisać program, w którym zostaną zdefiniowane 3 zmienne a, b, c oraz wskaźnik wsk (odpowiedniego typu) do wskazywania na te zmienne. Należy:
   - wyświetlić adresy zmiennych a, b, c za pomocą operatora referecji oraz wskaźnika;
   - zmienić wartości zmiennych a, b, c za pomocą wskaźnika (z wykorzystaniem operatora dereferencji).

2. Napisać i uruchomić program:
   a. wyznaczający średnią wartość elementów pewnej tablicy a ($\mu = \frac{\sum_i a_i}{n}$, gdzie μ = średnia, n = lb. elem. tablicy) ;
   b. obliczający wartość odchylenia standardowego dla elementów tablicy a ($\sigma = \sqrt{\frac{\sum_i (a_i - \mu)^2}{n}}$; σ= odchylenie std.);
   c. zliczający i zapisujący elementy o wartościach z przedziału $(\mu - \sigma, \mu + \sigma)$ w nowej tablicy *b*;
   *d. umieszczający elementy (tablicy a) o wartości mniejszej od średniej po lewej stronie nowej tablicy *c*, elementy niemniejsze od średniej - po prawej stronie tablicy *c*.

W programie powinny być zdefiniowane następujące funkcje:
- funkcja tworząca tablicę: tworzy dynamiczną tablicę o rozmiarze zadanym przez użytkownika, zwraca wskaźnik do utworzonej tablicy (ew. generuje wartości elementów tablicy);
- funkcja dla punktu a: przyjmuje 2 argumenty: wskaźnik do tablicy i lb. elem. tab., zwraca wartość średniej;
- funkcja dla punktu b: przyjmuje 3 argumenty: wskaźnik do tablicy, lb. elem. tab., wartość średniej, zwraca wartość odchylenia standardowego;
- funkcja dla punktu c: przyjmuje 4 argumenty: wskaźnik do tablicy, lb. elem., średnia, odchyl. std, tworzy nową tablicę i zwraca do niej wskaźnik;
- funkcja dla punktu d: przyjmuje 3 argumenty: wskaźnik do tablicy, lb. elem., średnia, tworzy nową tablicę i zwraca do niej wskaźnik.
- funkcja dla wyświetlania zawartości tablicy;

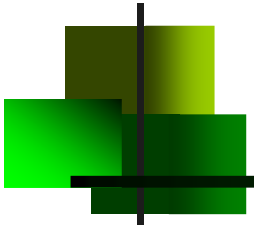
Zmiany wartości elementów tablic powinny odbywać się z wykorzystaniem operatora dereferencji *. Tablice powinny być definiowane w sposób dynamiczny (z wykorzystaniem słowa kluczowego new). Po zakończeniu pracy z tablicami należy zwolnić zajmowaną przez nie pamięć (delete).

2. Wskaźniki (2/2)

3. **Napisać program, w którym:**
   a. **zdefiniowane zostaną dwie dynamiczne tablice dwuwymiarowe;**
   b. **dla macierzy zapisanych w tych tablicach wykonane zostaną następujące działania:**
      a. **dodawanie, mnożenie, transpozycja;**

      (program powinien sprawdzać, czy posczególne działania mogą być wykonane);
   c. **zwolniona zostanie pamięć zajmowana przez tablice.**

* 4. **Napisać program, w którym:**
   a. **zostaną zdefiniowane funkcje zwieksz(float x, float d), zmniejsz(float x, float d) wykonujące, odpowiednio, zwiększenie i zmniejszenie wartości x o wartość d i zwracające zmienioną wartosć**
   b. **zdefiniowana zostanie funkcja dzialanie() wywołująca (za pomocą wskaźnika) funkcję zwieksz() lub zmniejsz(). Wskaźnik do funkcji zwieksz() lub zmniejsz() (użytkownik dokonuje wyboru działania) jest przekazywany jako argument do funkcji dzialanie().**



3. Przeciążanie funkcji

1. Napisać i uruchomić program, w którym zdefiniowanych zostanie 5 funkcji o nazwie obwod() obliczających i zwracających wartość obwodu następujących figur: koła, trójkąta, czworokąta, pięciokąta i sześciokąta. Długości boków figur mają być przyjmowane przez funkcje jako parametry.
2. Napisać i uruchomić program, w którym zdefiniowane zostaną 3 funkcje o nazwie wiekszy() wykonujące operację porównania dwóch przekazanych do nich argumentów i zwracających 1, gdy argument pierwszy ma większą wartość niż drugi, 2, gdy argument drugi ma większą wartość niż pierwszy, 0, gdy argumenty są sobie równe. Funkcje powinny działać dla argumentów będących liczbami całkowitymi, rzeczywistymi (z dokładnością do 0.001) i łańcuchami znaków.

4. Klasa, obiekt

1. Zdefiniować klasę Zamowienie reprezentującą informacje o zamowieniu. Klasa ma zawierać:
   - dane prywatne: nazwa_produktu, cena_produktu, data_zakupu, liczba_sztuk;
   - publiczne metody:
     - zapisz_dane() – zapisująca (do pól obiektu) dane o zamówieniu;
     - wyswietl_dane() – wyświetlająca dane o zamówieniu;
     - podaj_nazwe_produktu() – zwracająca wartość pola nazwa_produktu;
     - podaj_cene() – zwracająca wartość pola cena_produktu;
     - podaj_date_zakupu() – zwracająca wartość pola data_zakupu;
     - podaj_liczbe_sztuk() – zwracająca wartość pola liczba_sztuk;
     - podaj_koszt() – obliczająca i zwracająca całkowity koszt zamówienia.
2. Napisać i uruchomić program, definiujący dwa obiekty klasy Zamowienie, pobierający dane o dwóch zamówieniach, wypisujący informacje o tych zamówieniach i ich całkowite koszty.
3. Napisać i uruchomić program, który definiuje tablicę obiektów klasy Zamówienie oraz:
   a) zapamiętuje dane o kilku zamówieniach w tablicy,
   b) oblicza łączną wartość zamówień dla wskazanego produktu,
   c) oblicza łączną wartość zamówień złożonych we wskazanym miesiącu (można zastosować model daty zredukowany do miesięcy),
   d) wyświetla informacje o produkcie, który został zakupiony w największej ilości; uwzględnić sytuacje, gdy:
      1. każde zamówienie dotyczy innego produktu,
      2. dany produkt może wystąpić w więcej niż jednym zamówieniu.

5. Konstruktor, destruktor

1. **Dla programu z ćw 4. dodać definicje dwóch konstruktorów dla nadawania wartości danym tworzonych obiektów: a) konstruktor bez argumentów, b) konstruktor przyjmujący 4 argumenty. Zdefiniować nowe obiekty wykorzystując te konstruktory.**

2. **Napisać program definiujący klasę *Stozek*, reprezentującą stożek o nast. składowych:**
   - **prywatne pola (dane): *h* (wysokość stożka), *r* (promień podstawy), *l* (tworząca);**
   - **prywatna metoda: *oblicz_tworzaca()* – obliczająca i przypisująca polu *l* wartość tworzącej stożka;**
   - **dwa publiczne kontruktory nadające wartości polom obiektu**
     - **konstruktor bez parametrów, nadający danym wartosci zerowe,**
     - **konstruktor przyjmujący dwa parametry: wysokość *h* i promień *r*; (konstruktor ten powinien wywoływać metodę *oblicz_tworzaca()* dla nadania wartości polu *l*):**
   - **dwie metody publiczne obliczające i zwracające pole powierzchnni całkowitej oraz objętość stożka.**

   Zdefiniować obiekty klasy Stozek wykorzystując jej kontruktory. Obliczyć i wyświetlić pola powierzchni całkowitej i objętości stożków, reprezentowanych przez zdefiniowane obiekty.

2. ***Zdefiniować klasę *Wielokat* reprezentującą wielokąt. Składowe klasy:**
   - **prywatne: *n* (liczba wierzchołków wielokąta), *w* (adres początku tablicy (dynamicznej) przechowującej współrzędne wierzchołków - pojedynczy wierzchołek może być reprezentowany przez obiekt klasy reprezentującej punkt)**
   - **publiczne: konstruktor inicjalizujący tablicę *w* o rozmiarze *n*, konstruktor kopiujący, destruktor (zwalniający pamięć zajmowaną przez tablicę), metody wykonujące wybrane działania na wielokącie;**

   Zdefiniować dwa obiekty klasy *Wielokat* i wykonać dostępne dla niego działania (w tym kopiowanie wartości z jednego obiektu do drugiego).

6. Zapis i odczyt do/z plików

a) Zdefiniować następujące klasy:
   - Sprzet – dane prywatne: identyfikator, typ (rower, hulajnoga, rolki);
   - Osoba – dane prywatne: identyfikator, nazwisko, wiek;
   - Wypozyczenie - dane prywatne: identyfikator sprzętu, identyfikator osoby;

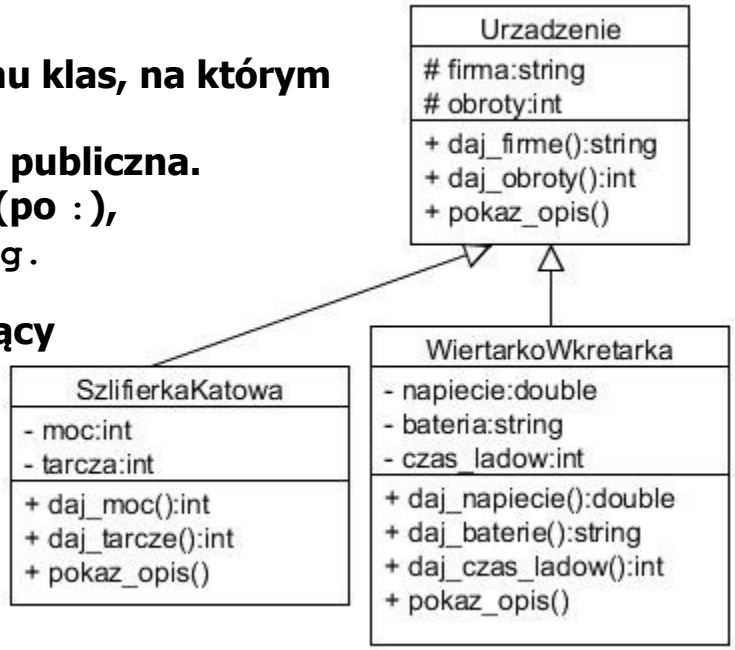
   Wszystkie klasy powinny zawierać publiczne konstruktory przypisujące danym wartości początkowe , publiczne metody wyświetlające wartości danych oraz publiczne metody udostępniające dane.

b) Napisać program, kóry:
   1) zapisuje do plików dane zawarte w wielu obiektach powyższych klas zapamiętanych w tablicach;
   2) odczytuje dane z plików i zapisuje je do tablic obiektów;
   3) wyświetla listę osób (nazwiska) w wieku powyżej 25 lat, które wypożyczyły hulajnogę;
   4) wyświetla listę osób (nazwiska) w wieku poniżej 10 lat, które wypożyczyły rolki;
   5) oblicza i wyświetla liczbę wypożyczonych rowerów;
   6) oblicza i wyświetla liczbę wypożyczonych par rolek.

7. Dziedziczenie

1. **Uruchomić program z przykładu 1 z wykładu "Dziedziczenie" (sl. 7), modyfikując definicję konstruktorów z parametrami dla klas pochodnych Sosobowy i SCiezarowy tak, aby na ich listach inicjalizacyjnych znajdowało się wywołanie konstruktora z parametrem z klasy podstawowej Samochod (przykład 2, sl. 10). (Diagramy klas dla przykładu 1 znajdują się na slajdach 12 i 13 wykładu).**

2.a) Zdefiniować hierarchię klas według poniższego diagramu klas, na którym zastosowano następujące oznaczenia:
– składowa prywatna, # składowa chroniona, + składowa publiczna.
Dla metod podane są typy zwracanych przez nie wartości (po :), dla pól podane są ich typy, np. pole `firma` jest typu `string`.



Dla każdej klasy należy zdefiniować konstruktor przyjmujący parametry (zadaniem konstruktora jest nadanie wartości wszystkim polom obiektu klasy), przy czym na liście inicjalizacyjnej konstruktora każdej z klas pochodnych należy umieścić wywołanie konstruktora z parametrami klasy podstawowej w celu nadania wartości polom odziedziczonym: `firma, obroty`.

b) Zdefiniować dwie tablice do przechowywania obiektów klas `SzlifierkaKatowa` i `WiertarkoWkretarka`, zapamiętać w tych tablicach dane dla kilku obiektów każdej z klas oraz wyświetlić dane o wiertarko-wktrętarce z baterią o najkrótszym czasie ładowania oraz dane o szlifierce o największej liczbie obrotów. Wszystkie dane dla danego urządzenia są pokazywane przez metodę `pokaz_opis()`.

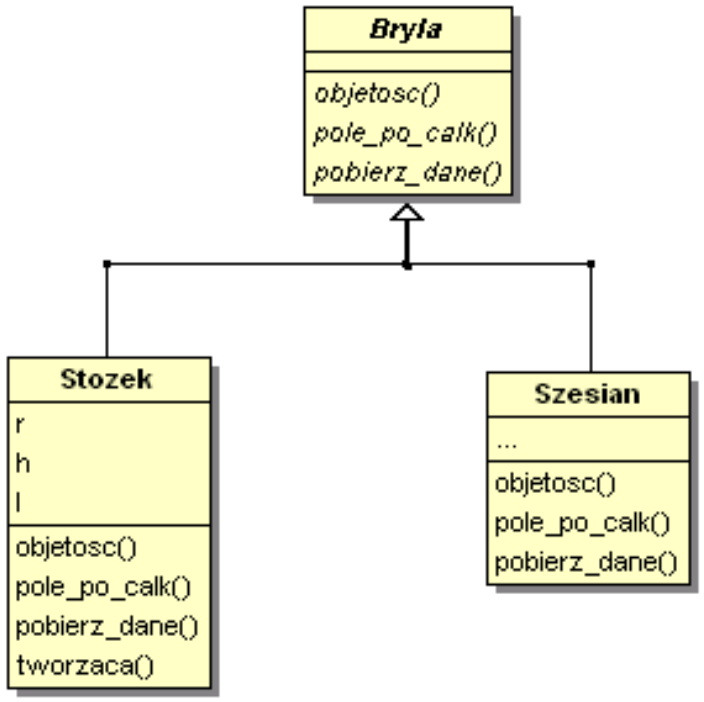
Znaczenie pól: `firma` – producent, `obroty` - liczba obrotów na minutę, `tarcza` – średnica tarczy [mm], `bateria` – typ baterii, `czas_ladow` – czas ladowania baterii [min]

8. Klasy abstrakcyjne, polimorfizm

1. Zaimplementować hierarchię, której fragment jest przedstawiony na rysunku, gdzie
   - `Bryla`: jest **klasą abstrakcyjną**, definiującą interfejs składający się z czysto wirtualnych metod: **`objetosc(), pole_powierzchni_calkowitej(), pobierz_dane()`**
   - **`Stozek, Walec, Kula, CzworoscianForemny, Prostopadloscian`** `i inne` są konkretnymi klasami pochodnymi klasy **`Bryla`**, zawierającymi implementacje metod czysto wirtualnych z klasy **`Bryla`** oraz pewne specyficzne dla siebie pola (dane) i metody prywatne
2. Wykonać działania dla obiektów klas pochodnych: wykorzystać **wskaźnik typu klasy bazowej `Bryla`** i **referencję typu klasy bazowej `Bryla`** (wywołania polimorficzne).

   Przykładowy fragment implementacji :

```
1.  void obsluga_bryly(Bryla&b){
2.    b.pobierz_dane();
3.    //…
4.  }
5.  int main(){
6.    Bryla* wsk= new Stozek;
7.    wsk->pobierz_dane();
8.    //…
9.    delete wsk;
10.   Stozek b;
11.   obsluga_bryly(b);
12. }
```



Ćwiczenia 9 - Przeciążanie operatorów

1. **Napisać program w którym:**
   - zdefiniowana zostanie klasa reprezentująca wektor
     - w klasie tej zostaną zdefiniowane operatory + , * ,- ,/ , <<
     - rozważyć dwa przypadki:
       - wektor jest tablicą statyczną
       - wektor jest tablicą dynamiczną
   - Uwagi:
     - można dodatkowo przeciążyć operatory arytmetyczne tak, aby jednym z argumentów operacji arytmetycznej była liczba (skalar.)
     - w przypadku tablicy tworzonej dynamicznie należy zdefiniować również konstruktor kopiujący, destruktor i operator przypisania kopiującego


10.