
Zadanie 1 - Funkcja minimum

Napisz szablon funkcji, o nazwie `minimum`, znajdującej najmniejszą wartość w tablicy.
Funkcja otrzymuje tablicę w postaci `vector` i zwraca znalezioną wartość.

```cpp
template<typename T>
T minimum(const std::vector<T>& tab) {
    T minimum = tab[0];
    for (size_t i = 1; i < tab.size(); ++i) {
        if (tab[i] < minimum) {
            minimum = tab[i];
        }
    }
    return minimum;
}
```

Zadanie 2 - Funkcja szukaj_auta

Dana jest klasa Auto.
Obiekty klasy Auto posiadają trzy publiczne pola typu string: marka, model i kolor.
Napisz funkcję szukaj auta() z trzema parametrami:
pierwszym jest vector obiektów klasy Auto,
drugim string
a trzecim wskaźnik na pole składowe typu string klasy Auto.

Funkcja ma wyszukać w wektorze pierwsze wystąpienie obiektu, który we wskazywanym polu zawiera podany string.
Funkcja ma zwracać indeks znalezionego obiektu.

```cpp
int szukaj_auta(
        const vector<Auto>& Auta,
        const string& wartosc,
        const string Auto::* pole) {       //pod *(marka,kolor,model)

    for (size_t i = 0; i < Auta.size(); ++i) {
        if (Auta[i].*pole == wartosc) {
            return static_cast<int>(i);
        }
    }
    return -1;
}
```

Zadanie 3 - Klasa Figura i dziedziczenie

Dane są klasy Koło i Kwadrat, które posiadają publiczną metodę pole() zwracającą pole danej figury jako liczba rzeczywista typu double (klasy są już zdefiniowane łącznie z metodami pole).

Zadaniem jest napisanie klasy Figura (będącej interfejsem), z której dziedziczą klasy Koło i Kwadrat.
Klasa ma posiadać czysto abstrakcyjną metodę pole(), która jest nadpisywana w klasach pochodnych.
```cpp
class Figura {
public:
    virtual double pole() const = 0;
    Figura(){}
    virtual ~Figura() = default;
};
```

Zadanie 4 - Funkcje dzielenia i serwis

Napisz dwie funkcje.
Pierwsza niech dostaje dwie liczby typu double i zwraca wynik dzielenia drugiej przez pierwszą.
Jeśli pierwsza liczba wynosi zero, funkcja ma rzucić wyjątek std::runtime_error z komunikatem "Dzielenie przez zero".
Druga funkcja ma nazywać się serwis i ma mieć takie same parametry jak pierwsza, ale nie ma nic zwracać.
Ma wywołać pierwszą funkcję i wypisać jej wynik oraz obsłużyć rzucony wyjątek.
Obsługa wyjątku ma polegać na wypisaniu komunikatu z obiektu wyjątku.
```cpp
double Dzielenie(double wartosc1, double wartosc2) {
            if ( wartosc1 == 0) {
                throw std::runtime_error("Dzielenie przez zero");
            }else {
                return wartosc2 / wartosc1;
            }
}
void serwis(double wartosc1, double wartosc2) {
    try {
        double wynik = Dzielenie(wartosc1, wartosc2);
        cout << "Wynik: " << wynik << endl;
    } catch (const std::runtime_error& e) {
        cout << "Blad: " << e.what() << endl;
    }
}
```

Zadanie 5 - Klasa Temperatura

Stwórz klasę Temperatura, która będzie posiadała pole prywatne stopnie typu double do przechowywania temperatury w skali Kelvina.
W klasie stwórz konstruktor, który otrzymuje wartość double traktując jako temperatura w skali Celsjusza i konwertuje ją do Kelvinów zapisując po konwersji w prywatnym polu stopnie.
Dodaj do klasy operator double, konwertujący obiekt Temperatura na wartość double będącą wartością temperatury w Celsjuszach.
Dla przypomnienia: 0 K = -273,15ºC.
Dla celów testowych dodaj do klasy metodę print, która wypisze na ekran wartość pola stopnie (samą wartość, bez żadnych dodatkowych tekstów).
```cpp
```

```cpp
class Temperatura {
private:
    double stopnie;
public:
    Temperatura(double stopnieC) {
        stopnie = stopnieC + 273.15;
    }
    ~Temperatura() {}
    operator double() const {
        return stopnie - 273.15;
    }
    void print() const {
        cout << stopnie << endl;
    }
};
```

Zadanie 1 - Klasa Transport i dziedziczenie

Stwórz klasę Transport z metodą wypisz(), klasy Samochod i Statek dziedziczące z Transport, oraz klasę Amfibia dziedziczącą po Samochod i Statek. Napisz funkcję transport() wypisującą odpowiednie komunikaty.

```cpp

#include <iostream>
class Transport {
public:
    virtual void wypisz() const {
        std::cout << "Jestem srodkiem transportu" << std::endl;
    }
    virtual ~Transport() = default;
};
class Samochod : virtual public Transport {
public:
    void wypisz() const override {
        std::cout << "Jestem samochodem" << std::endl;
    }
};
class Statek : virtual public Transport {
public:
    void wypisz() const override {
        std::cout << "Jestem statkiem" << std::endl;
    }
};
class Amfibia : public Samochod, public Statek {};
void transport() {
    Amfibia amfibia;
    amfibia.Samochod::wypisz();
    amfibia.Statek::wypisz();
    amfibia.wypisz();
}
```

Zadanie 2 - Szablon klasy Stos

Napisz szablon klasy Stos z metodą push() na wektorze.

```cpp
#include <vector>

template <typename T>
class Stos {
private:
   std::vector<T> arr;
public:
   void push(const T& element) {
      arr.push_back(element);
   }
};
```

Zadanie 3 - Konstruktor przenoszący i get_vector

Dana jest klasa, której niekompletna deklaracja znajduje się poniżej :
Dopisz konstruktor przenoszący dla klasy Tablica oraz metodę get_vector().

```cpp
   // Konstruktor przenoszący
   Tablica(Tablica&& other) noexcept : tab(other.tab), size(other.size) {
      other.tab = nullptr;
      other.size = 0;
   }

   // Metoda get_vector
   std::vector<int> get_vector() const {
      return std::vector<int>(tab, tab + size);
   }
};
```

Zadanie 4 - Własna klasa wyjątków i funkcje

Napisz klasę wyjątków dziedziczącą z std::exception oraz funkcje obliczającą pierwiastki i obsługującą wyjątki.

```cpp
#include <stdexcept>
#include <vector>
#include <cmath>
#include <iostream>

class MojWyjatek : public std::exception {
private:
   std::string komunikat;
   int pozycja;
public:
   MojWyjatek(const std::string& msg, int pos) : komunikat(msg), pozycja(pos) {}
   const char* what() const noexcept override {
      return komunikat.c_str();
   }
   int getPozycja() const {
      return pozycja;
   }
};

std::vector<double> pierwiastki(const std::vector<double>& liczby) {
   std::vector<double> wynik;
   for (size_t i = 0; i < liczby.size(); ++i) {
      if (liczby[i] < 0) {
         throw MojWyjatek("Pierwiastek z wartości ujemnej", i);
      }
      wynik.push_back(std::sqrt(liczby[i]));
   }
   return wynik;
}

void serwis(const std::vector<double>& liczby) {
   try {
      std::vector<double> wynik = pierwiastki(liczby);
      for (double x : wynik) {
         std::cout << x << " ";
      }
      std::cout << std::endl;
   } catch (const MojWyjatek& e) {
      std::cout << e.what() << " na pozycji " << e.getPozycja() << std::endl;
   }
}
```

Zadanie 5 - Klasa Zwierze i dziedziczenie

Stwórz klasę Zwierze z metodą wypisz(), klasy Ssak i Ryba dziedziczące z Zwierze, oraz klasę Delfin dziedziczącą po Ssak i Ryba. Napisz funkcję zwierzak() wypisującą odpowiednie komunikaty.

```cpp
#include <iostream>
class Zwierze {
public:
    virtual void wypisz() const {
        std::cout << "Jestem zwierzeciem" << std::endl;
    }
    virtual ~Zwierze() = default;
};
class Ssak : virtual public Zwierze {
public:
    void wypisz() const override {
        std::cout << "Jestem ssakiem" << std::endl;
    }
};
class Ryba : virtual public Zwierze {
public:
    void wypisz() const override {
        std::cout << "Plywam" << std::endl;
    }
};
class Delfin : public Ssak, public Ryba {};
void zwierzak() {
    Delfin delfin;
    delfin.Ssak::wypisz();
    delfin.Ryba::wypisz();
    delfin.wypisz();
}```
---
```