Podstawy programowania: Laboratorium nr 6

Struktury. Pliki nagłówkowe

18 listopada 2016

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. umieszczeniu wszystkich rozwiązań w jednym pliku, chyba, że w poleceniu napisano inaczej.
5. w zadaniach wymagających udzielenia komentarza bądź odpowiedzi, należy umieścić go w kodzie programu (np. w postaci komentarza albo wydrukować na ekranie).

Wprowadzenie

Struktury

W C++ struktury pozwalają na grupowanie powiązanych ze sobą informacji pod jedną nazwą. Przykładowo, aby przechowywać informacje o punktach na płaszczyźnie moglibyśmy zdefiniować strukturę `Point` w następujący sposób:

```
struct Point {
    double x;
    double y;
}
```

Do pól składowych instancji struktury można odnosić z wykorzystaniem operatora '.'. Przykład:

```
    Point p1; // deklaracja nowego punktu
    p1.x = 3.0;
    p1.y = 7.5;
    cout << p1.x << " " << p1.y << endl;    // wyświetli na ekranie: 3 7
```

Ze zdefiniowanej w ten sposób struktury można korzystać podobnie jak z innych typów. Przykładowo, funkcja wyświetlająca na ekranie zawartość podanego punktu mogłaby wyglądać następująco:

```
    void diplay(const Point& p) {
        cout << p.x << " " << p.y;
    }
```

Dodatkowe informacje:

- struktury - http://cplusplus.com/doc/tutorial/structures/

Pliki nagłówkowe

W przypadku rozbudowanych programów (kod źródłowy ciągle się rozrasta) można zauważyć, że zarządzanie kodem jest coraz trudniejsze - zwłaszcza, gdy wszystko jest implementowane w jednym pliku źródłowym. Rozwiązaniem tego problemu jest podzielenie kodu na wiele modułów, gdzie każdy moduł w pełni realizuje określone funkcje. Moduł można z kolei podzielić na dwie części: implementacja funkcjonalności (pliki *.cpp) oraz interfejs modułu (pliki nagłówkowe *.h, *.hpp). O ile pliki *.cpp zawierają implementacje funkcji i metod, o tyle plik nagłówkowy zawiera tylko deklaracje funkcji, typów, struktur (wraz z metodami), itp.

Przykładowy program o budowie modułowej może mieć następującą strukturę:

- matrix.cpp, matrix.hpp - moduł odpowiedzialny za działania na macierzach,
- studenci.cpp, studenci.hpp - moduł odpowiedzialny za operacje na liście studentów,
- pracownicy.cpp, pracownicy.hpp - moduł odpowiedzialny za operacje na liście pracowników,
- main.cpp - program główny, który wykorzystuje pozostałe moduły i integruje je w całość.

Pliki nagłówkowe załącza się w programie za pomocą znanej już dyrektywy `#include "nazwa.hpp"` (uwaga o ile załączanie plików *.cpp jest poprawne z punktu widzenia kompilatora, to w ogólnosci nie powinno się tego robić). Aby wykorzystać w module funkcję implementowane w innym module wystarczy załączyć odpowiedni plik nagłówkowy.

Ponieważ pliki nagłówkowe mogą być załączone wielokrotnie w całym programie istnieje niebezpieczeństwo, że jakaś funkcja, typ zostanie przypadkowo zadeklarowana wiele razy. Aby temu zapobiec w plik nagłówkowy powinien mieć następującą postać:

```
#ifndef NAZWA_PLIKU_HPP
#define NAZWA_PLIKU_HPP
/*
     zawartość pliku nagłówkowego
*/
#endif /* NAZWA_PLIKU_HPP */
```

Alternatywnie, możesz skorzystać z dyrektywy `pragma` w następujący sposób:

```
#pragma once
/*
     zawartość pliku nagłówkowego
*/
```

Dodatkowe informacje:

- dołączanie plików źródłowych – `http://en.cppreference.com/w/cpp/preprocessor/include`
- pliki nagłówkowe – `http://www.cplusplus.com/forum/articles/10627/`

Zadania

Zadanie 1

Zaprojektuj strukturę `Person` przechowującą informacje osobowe takie jak:

- `string firstName` – imię,
- `string lastName` – nazwisko,
- **`unsigned short`** `age` – wiek,
- **`char`** `gender` – płeć.

Definicję struktury umieść w oddzielnym pliku nagłówkowym, np. `person.hpp`.

W swoim programie, utwórz dwie zmienne typu `Person`, wypełnij je danymi a następnie wyświetl zawartość tych zmiennych na ekranie.

Zadanie 2

Zdefiniuj dwie funkcje:

a) `Person createPerson()` – do tworzenia nowej zmiennej typu `Person` z wykorzystaniem danych wczytywanych z klawiatury, oraz

b) **`void`** `display(`**`const`** `Person& p)` – do wyświetlania na ekranie zawartości struktury `Person`.

Deklaracje funkcji umieść w tym samym pliku nagłówkowym co poprzednio zdefiniowana struktura, natomiast implementację umieść w oddzielnym pliku źródłowym (np. `person.cpp`).

Przetestuj swoją implementację wykorzystując kolejno obie zaimplementowane funkcje.

Zadanie 3

Zaprojektuj strukturę `Student` przechowującą informacje: `imię`, `nazwisko`, `datę urodzenia`, oraz `numer indeksu`. Napisz funkcję o nazwie `display` wyświetlającą na ekranie zawartość struktury `Student`. Definicję struktury oraz implementacje funkcji umieść w osobnych plikach (np. w `student.hpp` oraz `student.cpp`).

W swoim programie, utwórz przykładową zmienną typu `Student` oraz wyświetl informacje w niej zawarte z wykorzystaniem nowo zdefiniowanej funkcji. Jak myślisz, w jaki sposób kompilator rozpoznaje, którą funkcję `display` ma wybrać w danym momencie (zdefiniowaną w ramach zadania 2 oraz w tym zadaniu), skoro obie posiadają tę samą nazwę?

Zadanie 4

Napisz program wczytujący n zmiennych typu `Person` do listy (n wczytaj od użytkownika). Następnie zapisz zawartość całej listy do wskazanego pliku tekstowego (np. `persons.txt`). Sprawdź zawartość pliku wynikowego – czy jesteś w stanie jednoznacznie odczytać wszystkie informacje?

Wskazówka 1 W celu zapisania listy osób skorzystaj z `std::vector`.
Wskazówka 2 W pliku tekstowym informacje o poszczególnych osobach możesz umieścić w osobnych liniach, natomiast dane pojedynczej osoby możesz rozdzielić spacją.

Zadanie 5

Napisz program wczytujący do listy zawartość całego pliku tekstowego, który utworzyłeś w poprzednim zadaniu. Następnie, wyświetl zawartość całej wczytanej listy oraz średnią wieku wszystkich osób.

Wskazówka Do wyświetlania osób skorzystaj z przygotowanej wcześniej funkcji `display`.

Zadanie 6

Wykorzystując algorytm sortowania bąbelkowego z poprzedniego laboratorium posortuj obiekty typu `Person` w tablicy względem wieku (możesz wczytać dane z pliku z poprzednich zadań). Wyświetl wynik – posortowaną tablicę osób. Czy algorytm działa poprawnie?

Na następne zajęcia

- Obsługa plików w trybie binarnym – `http://www.cplusplus.com/reference/fstream/`