Język C++ zajęcia nr 8

Składowe statyczne

Zwykłe dane składowe są niezależnie przechowywane w postaci odrębnych kopii w każdym obiekcie. Statyczne dane składowe przechowywane są w postaci jednej kopii należącej do definicji klasy. Składowe statyczne mogą mieć formę statycznych danych składowych i statycznych funkcji składowych.

Dostęp do składowych statycznych jest uzyskiwany na ogólnych zasadach dostępu do składowych obiektu lub przez użycie nazwy składowej **kwalifikowanej nazwą klasy**.

Statyczne dane składowe. Dana statyczna jest umieszczona w pamięci jako jedna tylko kopia wspólna dla wszystkich obiektów danej klasy. Przed jej deklaracją występuje modyfikator **`static`**. Deklaracja ta nie jest definicją. Definicja statycznej danej składowej, już bez powtarzania słowa **`static`,** musi wystąpić w takim miejscu programu, aby miała zasięg pliku. Definicja statycznej danej składowej może być połączona z jej inicjalizacją. Do zdefiniowanej składowej statycznej można odwołać się poprzez zapis:

nazwa_klasy* :: *nazwa_danej

Uwaga: Klasa definiowana lokalnie (w bloku lub w bloku funkcyjnym) nie może mieć statycznych danych składowych.

Statyczna funkcja składowa. Każda funkcja składowa, która operuje jedynie statycznymi danymi składowymi (nie licząc dostępu do niestatycznych składowych poprzez jawne podanie obiektu), nazywana jest statyczną funkcją składową. Przed deklaracją takiej funkcji występuje słowo **`static`**. Jeżeli definicja statycznej funkcji składowej występuje poza klasą, to nie może być powtarzane słowo **`static`**. **W statycznej funkcji składowej nie jest dostępny wskaźnik `this`**. Statyczna funkcja składowa może być aktywowana na rzecz całej klasy poprzez zapis:

nazwa_klasy* :: *nazwa_funkcji (lista_argumentów)

Statyczną funkcję składową można wywołać w standardowy sposób z podaniem obiektu, który służy w tym przypadku jedynie do określenia klasy, do której dana funkcja składowa należy (wartość wyrażenia określającego tę klasę nie jest wyliczana w trakcie wykonania programu, a jedynie podlega analizowaniu w procesie kompilacji).

ZINTERPRETUJ POSZCZEGÓLNE ELEMENTY KODU ŹRÓDŁOWEGO!!!

Dane i funkcje statyczne:

```
#include <iostream>
#include <string.h>
using namespace std;

class pracownik
{
  private:
    int wiek;
    static int preferowany_wiek;    // składowa statyczna, deklaracja !

  public:
    static int suma_lat;            // składowa statyczna, deklaracja !
    static int liczba_osob;         // składowa statyczna, deklaracja !
    pracownik(int w);               // konstruktor
    pracownik();                    // konstruktor domniemany
    static void raport();           // funkcja statyczna deklaracja !
    static void ustal_wiek(int prwk);    // funkcja statyczna deklaracja!
};

//------------------- Konstruktory ---------------------------------

pracownik::pracownik(int w)
{
  wiek = w;
  liczba_osob ++;
  suma_lat += wiek;
}

pracownik::pracownik()
{
  wiek = preferowany_wiek;
  cout << "\n---> Konstruktor domniemany, przyjeto wiek " <<
wiek;
  liczba_osob ++;
  suma_lat += wiek;
}
```

```
//------------------- Definicje funkcji składowych statycznych ------

void pracownik::raport()
{
  cout << "\nRaport: liczba osob " << liczba_osob <<
          ", srednia wieku ";
  if(liczba_osob != 0) cout << suma_lat/liczba_osob;
  else cout << "nieokreslona";
}

void pracownik::ustal_wiek(int prwk)
{
  preferowany_wiek = prwk;
}

//------------------- Definicje danych składowych statycznych -------
int pracownik::preferowany_wiek = 40;
int pracownik::suma_lat = 0;
int pracownik::liczba_osob = 0;
//---------------------------------------------------------------

int main()
{
  pracownik::raport();          // wywołanie funkcji przez nazwę klasy
  pracownik a(46), b(28);
  pracownik::raport();          // wywołanie funkcji przez nazwę klasy
  pracownik c,d;
  b.raport();                   // wywołanie funkcji przez nazwę obiektu
  pracownik::ustal_wiek(32);    // wywołanie funkcji przez nazwę klasy
  pracownik e,f,g;
  f.raport();                   // wywołanie funkcji przez nazwę obiektu
}
```

Oczekiwane wyniki:

```
Raport: liczba osob 0, srednia wieku nieokreslona
Raport: liczba osob 2, srednia wieku 37
---> Konstruktor domniemany, przyjeto wiek 40
---> Konstruktor domniemany, przyjeto wiek 40
Raport: liczba osob 4, srednia wieku 38
---> Konstruktor domniemany, przyjeto wiek 32
---> Konstruktor domniemany, przyjeto wiek 32
---> Konstruktor domniemany, przyjeto wiek 32
Raport: liczba osob 7, srednia wieku 35
```

Uwaga: Wywołanie funkcji statycznej poprzez operator składowej . lub -> **nie powoduje wyznaczenia wartości lewego argumentu** tego operatora. Argument ten określa jedynie klasę właściwą dla wywoływanej w ten sposób funkcji.

Konstruktor kopiujący

Inicjalizatorem obiektu pewnej klasy może być inny istniejący już obiekt tej klasy. W takim przypadku inicjalizacji tworzonego obiektu dokonuje zdefiniowany dla danej klasy **konstruktor kopiujący**, będący konstruktorem, który może być wywołany z jednym argumentem typu obiekt klasy lub referencja do takiego obiektu. W klasie **X** konstruktor kopiujący ma postać funkcji składowej:

X* :: *X (X&);

lub

X* :: *X (const X&);

Wywołanie konstruktora kopiującego następuje w przypadku:

- Definiowania nowego obiektu klasy wraz z inicjalizowaniem go wartością innego obiektu tej klasy:

```
X p;

X q(p);

X r = p;

X s = X(p);
```

- Przekazywania do funkcji argumentu będącego obiektem klasy.
- Zwracania przez funkcję rezultatu w formie obiektu klasy.

ZINTERPRETUJ POSZCZEGÓLNE ELEMENTY KODU ŹRÓDŁOWEGO!!!

Wykorzystanie konstruktorów kopiujących:

```
#include <iostream>
using namespace std;

class liczba
{
   public:
   int licz;
   liczba(int);     // deklaracja konstruktora jednoarg.
   liczba(liczba&); // deklaracja konstruktora kopiującego
   void drukuj();
};

liczba::liczba(int k)
{
   licz=k;
}

liczba::liczba(liczba& t)
{
   licz=t.licz;
   cout<<"---> Pracuje konstruktor kopiujacy"<<endl;
}

void liczba::drukuj()
{
   cout<<"liczba= "<<licz<<endl;
}

int main()
{
   liczba a(5);
   a.drukuj();
   liczba c=a;
   c.drukuj();
}
```

Oczekiwane wyniki:

```
liczba= 5

---> Pracuje konstruktor kopiujacy

liczba= 5
```

Część II – Program CPP08 – do realizacji podczas zajęć.

Napisz, skompiluj i uruchom program służący do operacji na kwadratach wg poniższych wytycznych:

- zdefiniuj klasę *kwadrat* zawierającą dane składowe typu double: *bok* i *pole* przechowujące odpowiednio długość boku kwadratu i jego pole;

- zdefiniuj konstruktor jednoargumentowy przypisujący swój argument składowej *bok* i wypisujący: komunikat „Utworzono kwadrat o boku”, długość boku i przejście do nowej linii;

- zdefiniuj zamkniętą (nie *inline*) funkcję składową klasy kwadrat *obliczpole*(), wyznaczającą wartość składowej *pole* i drukującą jej wartość (z odpowiednim opisem i przejściem do nowej linii); funkcja ta nie zwraca żadnej wartości;

- zdefiniuj destruktor wypisujący komunikat: „Usunieto kwadrat”;

- napisz funkcję main postaci:

```
void main()
{
   kwadrat k(3.5);
   k.obliczpole();
}
```

Część III – Ew. pytania i wyjaśnienia problemów dot. programu CPP07