Konstruktory, przeładowanie funkcji, funkcja inline

funkcja inline

- funkcja, której treść jest umieszczana w linijce, w której występuje wywołanie
- funkcje inline dla bardzo krótkich i małych funkcji, tylko wtedy mają sens

przeładowanie nazwy funkcji

- w danym zakresie ważności jest więcej niż 1 funkcja o takiej samej nazwie
- wywołanie danej funkcji zależy od liczby i typu argumentów wywołania
- (przy przeładowaniu ważna jest jedynie odmienność listy argumentów; nie jest brany pod uwagę zwracany typ)

lista inicjalizacyjna

- składnik nie const (zwykły) może mieć nadaną wartość początkową na dwa sposoby:
  1. przez pozycję na liście inicjalizacyjnej
  2. przez podstawienie wartości w ciele konstruktora

lista inicjalizacyjna cd.

- składnikowi z przydomkiem const można za pomocą konstruktora nadać wartość początkową TYLKO na liście inicjalizacyjnej konstruktora
- wybrana przez nas kolejność nie ma znaczenia dla kompilatora: inicjalizacja składników odbywa się w kolejności, w jakiej dane składniki są deklarowane w ciele klasy

KLASY

Klasa jest typem złożonym, składającym się z danych i funkcji (zwanych metodami).

Deklaracja i definicja

```
class NazwaTypu
{
 deklaracje danych i funkcji;
};
```

Deklaracja jest zapowiedzią klasy i polega na przytoczeniu jej ustroju w klamrach umieszczonych za słowem kluczowym `class`. Definicja klasy oznacza definicję jej funkcji. Definicje można przytaczać bezpośrednio w deklaracji, a także poza nią.

Przykład

Jedna funkcja zdefiniowana w deklaracji, druga poza nią:

```
class Wymierna
{
 private:
    int licznik, mianownik;
 protected:
 public:
    Wymierna(int l, int m){licznik =
    l; mianownik = m;};
    void wypisz(void);
};
void Wymierna :: wypisz(void)
{
 cout << licznik << "/" << mianownik;
}
```

Częsty błąd

- Funkcja definiowana poza klasą nie jest zaopatrzona w etykietę przynależności (tutaj `Wymierna ::`).

Funkcje przeciążone

Funkcje o identycznej nazwie, ale o zróżnicowanych typach zwracanych i/lub listach argumentów, są traktowane jak oddzielne algorytmy. Przeciążanie podnosi czytelność programów.

Przykład

```
int suma(int a, int b);
int suma(int a, int b, int c);
double suma(double a, double b);
...
a = suma(1, 2) + suma(1, 2, 3) +
suma(1.2, 1.3);
```

Częsty błąd

- Wywołanie funkcji z takim zestawem argumentów, że kompilator nie może jednoznacznie stwierdzić, który egzemplarz ma zostać wywołany.

Konstruktory klasy

```
class Nazwa
{
   Nazwa(lista argumentów);
   Nazwa(inna lista argumentów);
...
};
```

Funkcja o nazwie identycznej z nazwą klasy i niezwracająca rezultatu (nawet `void`). Zazwyczaj klasa ma kilka konstruktorów.

Przykład

```
class Punkt
{
 public:
   Punkt(void);
};
```

Lista inicjalizacyjna konstruktorów

Dane należące do klasy najwydajniej inicjalizujemy za pomocą listy. Elementy stałe `const` można zainicjalizować tylko przy użyciu listy. Elementów wspólnych dla wszystkich obiektów (`static`) nie wolno inicjalizować za pomocą listy.

Przykład

```
class Punkt
{
 private:
   int x, y;
   const int kolor;
 public:
   Punkt(int A, int B) : x(A), y(B),
kolor(RED){};
 };
```

Modyfikatory typów

const

Oznaczenie danej stałej. Dana stała powinna być zainicjowana w momencie deklaracji. Może być oznaczeniem wskaźnikowych lub referencyjnych rezultatów albo argumentów funkcji.

Przykład

```
const double pi = 3.14;
void drukuj(const Student &student);
```

static

Wskaźnik this

Wskaźnik do bieżącego egzemplarza klasy (obiektu). Wskaźnik `this` jest intensywnie eksploatowany podczas definiowania operatorów w klasach.

Przykład

```
class Wymierna
{
...
   void wypisz(void)
   { cout << this -> licznik << "/"
   << this -> mianownik;}
};
```

Dynamiczne deklarowanie zmiennych

```
Typ *adres = new Typ;
...
delete adres;
```

Dynamiczne tworzenie zmiennych określonego typu polega na zadeklarowaniu wskaźnika dla tego typu i utworzeniu pod jego adresem obszaru pamięci przeznaczonego na zmienną. Zmienna utworzona dynamicznie koniecznie musi być zlikwidowana, gdy nie jest już potrzebna.

Przykład

```
double *adres = new double;
*adres = 17.1;
cout << sin(*adres);
delete adres;
```

Częste błędy

- Brak słowa `delete` — czyli tzw. błąd wycieku pamięci.