

# Laboratorium Podstaw Informatyki

## 1. Przebieg ćwiczenia laboratoryjnego

### ***Przeładowanie operatorów***

Przeładowanie operatorów w języku C++ jest bardzo rozbudowanym mechanizmem pozwalającym na zwiększenie czytelności programu oraz znaczne uelastycznienie tworzonego kodu. Technika ta pozwala na korzystanie z typów danych zdefiniowanych przez użytkownika na równi z typami wbudowanymi. Na przykład dla zmiennych typu `int` można wykonać operacje:

```
int a, b;  
a = 3;  
b = a + 1;
```

Dzięki możliwości przeładowania operatorów, użytkownik który stworzy klasę liczb zespolonych może w kodzie programu zastosować konstrukcję:

```
zesp a(3,2);  
zesp b(2,0);  
zesp c;  
c = a + b;
```

Jak widać zapis ten jest bardzo intuicyjny i znacznie prostszy niż stosowanie specjalnej funkcji służącej do sumowania dwóch liczb zespolonych. W języku C++ możliwe jest przeładowanie wszystkich istniejących operatorów, z wyjątkiem: `.` `.*` `::` `?:`

Przeciążania operatorów dokonujemy za pomocą funkcji o specjalnych nazwach. Przykład deklaracji takiej funkcji:

```
typ operator+(typ a, typ b);
```

Funkcja ta przeciąża operator `+` dla obiektów klasy `typ`. Wywoływana jest automatycznie w sytuacji, gdy kompilator napotka wyrażenie:

```
typ ob1, ob2;  
typ wynik;  
wynik = ob1 + ob2;
```

Przeładować można operatory jednoargumentowe oraz dwuargumentowe. Jako jeden z argumentów przeładowanego operatora, użyty musi być obiekt klasy zdefiniowanej przez użytkownika. Jest to warunek konieczny, gdyż nie można przeładować operatorów dla typów wbudowanych. Takie operatory są już zdefiniowane i ich redefinicja nie ma sensu. Przeładowany operator może być zdefiniowany na dwa sposoby:

- Jako funkcja globalna
- Jako metoda składowa klasy

Różnica pomiędzy tymi dwoma sposobami polega na definicji danego operatora. Dla operatorów globalnych podajemy jawnie oba argumenty. Natomiast dla operatora w postaci metody składowej, możemy pominąć pierwszy argument (lub jedyny, dla operatorów jednoargumentowych). Ten pominięty argument zastępowany jest poprzez obiekt klasy, na rzecz której wywoływany jest operator. Różnica w definicji pokazana zostanie w przykładzie.

#### **Uwagi praktyczne:**

Mechanizm przeładowania operatorów daje programiście duże możliwości rozbudowy własnych klas. Istnieje jednak zagrożenie, iż niepoprawnie użyty mechanizm przeładowania operatorów spowoduje utrudnienie w czytaniu kodu programu. Z tego powodu warto pamiętać o kilku istotnych sprawach:

- Przeładowywać można tylko istniejące operatory i dopuszczone do przeładowania. Nie można tworzyć własnych, nowych operatorów.
- Nie można zmienić priorytetu operatorów ani ich łączności.
- Nie można zmienić ilości argumentów danego operatora.
- Przynajmniej jeden z argumentów operatora musi być typu zdefiniowanego przez użytkownika.
- Dobrym zwyczajem jest podobne działanie operatorów własnych i wbudowanych. To znaczy, jeżeli dany operator wbudowany nie modyfikuje argumentów, warto aby ten zdefiniowany przez użytkownika też tego nie robił.
- Jeżeli operator jest metodą składową klasy, to wymagane jest aby przy jego wywołaniu obiekt stojący po lewej stronie operatora, był obiektem tej klasy.
- Operatory `=` `[]` `()` oraz `->` muszą być zdefiniowane jako metody składowe klasy.
- Warto przeładowywać tylko te operatory które są konieczne. Jeżeli daną operację lepiej opisuje nazwa funkcji niż kształt operatora, lepiej zastosować funkcję.
- Jeżeli dla typów wbudowanych dany operator zwraca pewną wartość, warto aby dla typu własnego też to robił. Dzięki temu możliwe jest kaskadowe łączenie operatorów.

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

class wektor
{
public:
    int a;
    int b;
    wektor operator+(int arg);
};

wektor wektor::operator+(int arg)
{
    this->a = a + arg;
    this->b = b + arg;
    return *this;
}

wektor operator-(wektor wek, int licz)
{
    wek.a = wek.a - licz;
    wek.b = wek.b - licz;
    return wek;
}

int main(void)
{
    wektor w1, w2;
    w1.a = 5;
    w1.b = 8;
    w2 = w1 + 5;
    w1 = w2 - 3;
    cout << "1: A=" << w2.a << " " << "B=" << w2.b << endl;
    cout << "2: A=" << w1.a << " " << "B=" << w1.b << endl;
    getchar();
    return 0;
}
```

**Treść zadania laboratoryjnego:**

1. Utwórz klasę Punkt3D. W niej zaimplementuj 3 pola składowe x, y oraz z typu float.
2. Utwórz klasę Wektor, a niej zaimplementuj 3 pola składowe dx, dy, oraz dz typu float.
3. Zdefiniuj operator dodania do obiektu klasy Punkt3D liczby całkowitej int. Wszystkie współrzędne punktu, mają zostać zwiększone o wartość tej liczby.

4. Zdefiniuj operator przeładowany dla operacji dodania obiektu klasy Wektor do obiektu klasy Punkt3D, który do odpowiednich pól x, y oraz z w klasie Punkt3D przypisze odpowiednie przesunięcia z klasy Wektor.
5. Zdefiniuj operator mnożenia skalarnego (typu int) dla dwóch obiektów klasy Wektor.
6. W programie zdefiniuj 2 różne punkty oraz 2 wektory. Dokonaj translacji pierwszego punktu o pierwszy wektor. Następnie dokonaj mnożenia skalarnego obu wektorów. Na końcu do drugiego punktu dodaj wynik otrzymany z wykonanego wcześniej mnożenia dwóch wektorów.
7. Zdefiniuj operator <<, aby móc na ekran wypisywać w postaci strumieniowej położenie obiektu klasy Punkt3D.

Przykład:

```
Punkt3D a;  
a.x = 3;  
a.y = 5;  
a.z = 0;  
cout << a << endl; // Wypisze na ekran tekst: (3,5,0)
```