Laboratorium Informatyki II
Ćwiczenie 8

1 Przeciążanie funkcji

W języku `C` nazwa każdej funkcji musiała być unikalna, natomiast w `C++` możliwe jest zdefiniowanie więcej jak jednej funkcji o tej samej nazwie. Procedurę tą określa się jako przeciążenie funkcji i ma ona także zastosowanie do metod klasy. Przeciążone funkcje (metody) różnią się typem zwracanej wartości oraz liczbą i typami atrybutów. Przeciążenia nie mogą różnić się tylko zwracanym typem.

```
int suma(int,int);
float suma(float, float);
float suma(float,int);
int suma(float,int); //nie można zdefiniować takiego przeciążenia
int suma(float,int,int);
```

W klasie przeciążane są najczęściej metody konstruktora, zapewniając kilka sposobów na zainicjowanie obiektu klasy, jak pokazano poniżej.

```
#include <iostream>
class dane
{
private:
    float x;
    int n, *tab;
    std::string nazwa;
public:
    dane(){
        x = 0; n = 1;
        nazwa = "EMPTY";
        tab = new int[1];
        tab[0] = x;
    }
    dane(int x, int n)
    {
        nazwa = "FIRST";
        this->x = x;
        this->n = n;
        tab = new int[n];
        for(int i=0;i<n;i++)
            tab[i] = x+i;
    }
```

```
    dane(std::string nazwa, int n)
    {
        this->nazwa = nazwa;
        this->n = n;
        x = nazwa.length();
        tab = new int[n];
        for(int i=0;i<n;i++)
            tab[i] = x-i;
    }
    ~dane()
    {
        delete [] tab;
    }
    void drukuj()
    {
        std::cout << nazwa << " : ";
        for(int i=0;i<n;i++)
            std::cout << tab[i] << " ";
        std::cout << "| x=" << x << std::endl;
    }
};
int main()
{
    dane A, B("AMEBA",3), C(27,4);
    A.drukuj();
    B.drukuj();
    C.drukuj();
}
```

2 Przeciążanie operatorów

W podobny sposób można nadać nową funkcjonalność operatorom stosowanym w języku C++. W języku C++ można przeciążyć poniższe operatory:

1. Arytmetyczne:
   - + (dodawanie)
   - - (odejmowanie)
   - * (mnożenie)
   - / (dzielenie)
   - % (modulo)
2. Porównania:
   - == (równość)
   - != (nierówność)
   - < (mniejsze niż)
   - > (większe niż)
   - <= (mniejsze lub równe)

- `>=` (większe lub równe)

3. Przypisania:
   - `=` (przypisanie)
   - `+=` (dodawanie i przypisanie)
   - `-=` (odejmowanie i przypisanie)
   - `*=` (mnożenie i przypisanie)
   - `/=` (dzielenie i przypisanie)
   - `%=` (modulo i przypisanie)
4. inkrementacji i dekrementacji:
   - `++` (inkrementacja)
   - `--` (dekrementacja)
5. Bitowe:
   - `&` (bitowe AND)
   - `|` (bitowe OR)
   - `^` (bitowe XOR)
   - `~` (bitowe negacja)
   - `<<` (przesunięcie bitowe w lewo)
   - `>>` (przesunięcie bitowe w prawo)
6. Logiczne:
   - `!` (negacja logiczna)
   - `&&` (logiczne AND)
   - `||` (logiczne OR)
7. Indeksowania:
   - `[]` (indeksowanie)
8. Wywołania funkcji:
   - `()` (wywołanie funkcji)
9. Operator warunkowego przypisania:
   - `?:` (warunek ? wartość1 : wartość2)

Z wymienionych powyżej operatorów wyróżnia się "`!`" ponieważ jako jedyny nie wymaga zdefiniowania atrybutu. Przeciążenie operatora definiuje się zazwyczaj w klasie, ale można także w kodzie programu analogicznie jak deklarację funkcji. Przeciążenie deklaruje się za pomocą słowa kluczowego `operator` zgodnie ze składnią pokazaną poniżej.

```
typ_zwracany operator znak(typ paramert)
{
    kod;
    return wynik;
}
```

W pewnych sytuacjach (konfiguracjach wywołania operatora) nie jest możliwe zdefiniowanie danego działania w klasie. W takiej sytuacji możliwe jest zdefiniowanie przeciążenia operatora poza klasą w kodzie programu. Sytuacja taka przykładowo występuje gdy konieczne jest przeciążenie operatora dodawania realizującego sumę obiektu klasy i liczby całkowitej. Zaprogramowanie odwrotnego działania w kodzie klasy jest niemożliwe, i musi zostać zrealizowane poza klasą. W takim przypadku deklaracja wymaga podania dwóch atrybutów przeciążenia w nawiasie okrągłym, pierwszym będzie zmienna całkowita, a drugim obiekt danej klasy. Przykład implementacji przeciążenia pokazano w kodzie poniżej.

```
#include <iostream>
class dane
{
private:
    float x, y;
public:
    dane()
    {
        x = 0;y = 0;
    }
    dane(int x, int y)
    {
        this->x = x;
        this->y = y;
    }
    dane operator+(int x)
    {
        dane temp;
        temp.x = this->x + x;
        temp.y = this->y + x;
        return temp;
    }
    void drukuj()
    {
        std::cout << "(" << x << "," << y << ")" << std::endl;
    }
};
dane operator+(int x, dane Liczba)
{
    dane temp;
    temp=Liczba+x;
    return temp;
}
int main()
{
    int x = 5, y = 23;
    dane A(12,34), B(33,-23), C;
    C = A + x;
    C.drukuj();
    C = y + B;
    C.drukuj();
}
```

Zadania

W oparciu o informacje wykładowe i treść niniejszej instrukcji zmodyfikować program zbierający dane o przebiegu wycieczki w którym poszczególne zadania zostaną zrealizowane z wykorzystaniem przeciążonych operatorów. Zaproponować dodatkowe funkcjonalności programu z wykorzystaniem przeciążonych operatorów i funkcji (metod).