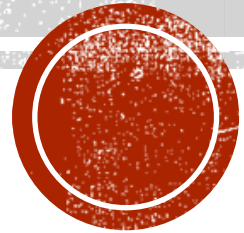
PRZECIĄŻENIE OPERATORÓW OBSŁUGA WYJĄTKÓW



Kamil Legierski

Wykorzystano materiały:
Vardan Grigoryan, Shunguang Wu, Expert C++. Packt, 2020.
https://www.udemy.com/course/programowanie-obiektowe-w-jezyku-cpp-oop-od-a-do-z/

AGENDA

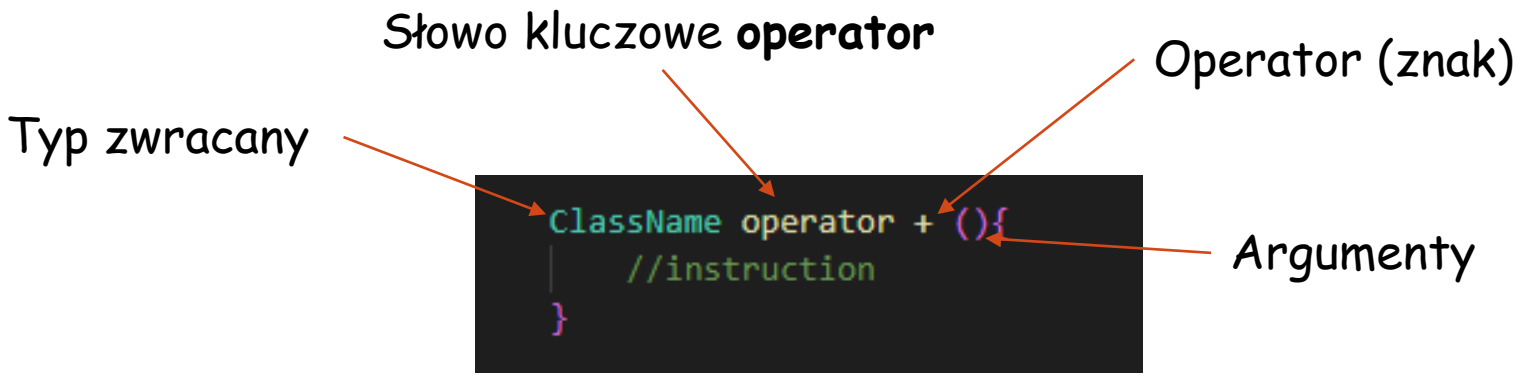
- Przeciążenie operatorów – co to ?
- Przeciążenie operatorów jednoargumentowych
- Przeciążenie inkrementacji i dekrementacji
- Przeciążenie operatorów logicznych
- Przeciążenie operatorów wejścia/wyjścia

PRZECIĄŻENIE OPERATORÓW

- Przy pomocy przeciążenia operatorów możemy nadać specjalne znaczenie wskazanym operatorom.

```
ClassName o1(2,3), o2(4,5);
ClassName o3 = o1 + o2;
```

SKŁADNIA


Słowo kluczowe **operator**
Operator (znak)
Typ zwracany
ClassName operator +(){
//instruction
}
Argumenty

ZASADY DOTYCZĄCE PRZECIĄŻANIA OPERATORÓW

- Przeciążone mogą być jedynie operatory wbudowane w C++
- Przeciążony operator nie może mieć parametrów domyślnych
- Nie można przeciążać operatorów dla wbudowanych typów danych
- Pierwszeństwo i asocjacyjność operatorów są nieznane
- Niektóre operatory (, , & , =) są przeciążone domyślnie
- Nie można zmienić ilości operatorów

JAKICH OPERATORÓW NIE PRZECIĄŻAMY ?

Operator	Znaczenie
::	Operator zasięgu
?:	Operator warunkowy
.	Wybór elementu członkowskiego
.*	Wybór wskaźnika do elementu członkowskiego
sizeof	Operator do sprawdzania rozmiaru

JAKICH OPERATORÓW PRZECIĄŻAMY ?

Operator	Znaczenie
!	Negacja logiczna
!=	Operacja różności
%	Reszta z dzielenia(modulo)
%=	Przypisanie reszty z dzielenia
&	Bitowa koniunkcja(AND) / Adres
&&	Iloczyn logiczny(AND)
&=	Przypisanie wyniku bitowej koniunkcji
()	Wywołanie funkcji / Operator rzutowania
*	Mnożenie / Odwołanie do wskaźnika
*=	Przypisanie wyniku mnożenia

Operator	Znaczenie
+	Dodawanie / Jednoargumentowy plus
++	Operator inkrementacji
+=	Przypisanie wyniku dodawania
-	Odejmowanie / Jednoargumentowy minus
--	Operator dekrementacji
-=	Przypisanie wyniku odejmowania
->	Pośredni operator przynależności
->*	Pośrednia dereferencja składowej
/	Dzielenie
/=	Przypisanie wyniku dzielenia

JAKICH OPERATORÓW PRZECIĄŻAMY ?

Operator	Znaczenie
<	Mniejsze niż
<=	Mniejsze lub równe
>	Większe niż
>=	Większe lub równe
<<	Przesunięcie w lewo
<<=	Przypisanie wyniku przesunięcia w lewo
>>	Przesunięcie w prawo
>>=	Przypisanie wyniku przesunięcia w prawo
=	Proste przypisanie
==	Równość

Operator	Znaczenie
[]	Indeks tablicy
^	Bitowa różnica symetryczna
^=	Przypisanie wyniku bitowej różnicy symetrycznej
|	Bitowa alternatywna
|=	Przypisanie wyniku bitowej alternatywy
||	Suma logiczna(OR)
new	Dynamiczna alokacja pamięci
delete	Dynamiczne zwolnienie pamięci
new[]	Dynamiczne alokowanie tablicy
delete[]	Dynamiczne zwolnienie tablicy
,	Połączenie dwóch wyrażeń w jedno

PRZECIĄŻENIE OPERATORÓW JEDNOARGUMENTOWYCH

- Jednoargumentowe - operatory jednoargumentowe (unarne) wiązane z jednym operandem

Operator	Znaczenie
+	Dodawanie
-	Odejmowanie
++	Inkrementacja
--	Dekrementacja
!	Logiczne NOT
&	Adres
*	Odwołanie do wskaźnika

PRZECIĄŻENIE INKREMENTACJI I DEKREMENTACJI

Operator	Znaczenie
++	Inkrementacja
--	Dekrementacja

PRZECIĄŻENIE OPERATORÓW LOGICZNYCH

- operatory relacyjne: >, >=, <, <= W wyniku działania tych operatorów otrzymujemy odpowiedź: prawda (true - 1) lub fałsz (false- 0). Priorytet tych operatorów jest taki sam.
- operatory przyrównania ==, != Ich priorytet jest niższy niż priorytet operatorów relacji
- operatory sumy i iloczynu logicznego, || - realizujący sumę logiczną (LUB - alternatywa), && - realizujący iloczyn logiczny (I - koniunkcja)

PRZECIĄŻENIE OPERATORÓW WEJŚCIA/WYJŚCIA

- Operatory wejścia/wyjścia mogą być przeciążone na typach danych zdefiniowanych przez użytkownika
- Operatory muszą być przeciążone jako funcja globalna.
- Prototyp przeciążenia wyjścia:

```
std::ostream &operator<<(std::ostream& out, const ClassName &obiekt);
```

- Prototyp przeciążenia operatora wejścia:

```
std::istream &operator>>(std::istream& in, ClassName &obiekt);
```

ZADANIE 1

- Napisz klasę Vec3d, która pozwoli na wykonywanie działań na wektorach. Twoja klasa ma pozwolić na operację:
  - Dodawania wektorów
  - Odejmowania wektorów
  - Inkrementacji wektorów
  - Mnożenia wektorów
  - Porównanie wektorów (równe, różne)
  - Wyświetlaj wektor w postaci (X: , Y: , Z:)
  - Pozwoli wprowadzić użytkownikowi wektor przy użyciu konsoli

*na potrzeby zadanie przyjmimy, że współrzędne to liczby całkowite.

AGENDA

- Wyjątek – co to ?
- Obsługa wyjątków
- Wyjątki w funkcji
- Wiele wyjątków
- Wyjątki z wykorzystaniem klas
- Wyjątki w bibliotece standardowej

WYJĄTEK

- Napotkany problem w programie w czasie wykonywanie

OBSŁUGA WYJĄTKÓW

- Do obsługi wyjątków potrzebne są 3 słowa kluczowe :
  - **try** – określa blok kodu, który może zgłosić wyjątek,
  - **throw** – służy do zgłoszenia wyjątku,
  - **catch** – określa blok kodu, który jest wykonywany w momencie zgłoszenia wyjątku

```
try{
    //blok kodu, ktory moze zglosic wyjatek
    throw; // wyrzuca wyjątek
}
catch(...){
    //blok kodu, wykonywany jesli wystapi wyjatek
}
```

WYJĄTKI W FUNKCJI

```
double divide(double i, double j){
    if(j == 0)
        throw std::string("Cannot divide by 0");
    return i/j;
}
```

```
try{
    divide(i,j);
}
catch(std::string& s){
   std::cout<<s;
}
```

WIELE WYJĄTKÓW

```
double divide(double i, double j){
    if(j == 0)
        throw std::string("Cannot divide by 0");

    if(i < 0){
        throw i;
    }
    if(j<0){
        throw j;
    }

    throw "nieoczekwiany wyjatek";

    return i/j;
}
```

```
try{
    divide(i,j);
}
catch(std::string& s){
   std::cout<<s;
}
catch(double e){
    std::cout << e << " jest mniejsze od 0";
}
catch(...){
    std::cout<<"Wystapil nieoczekiwany wyjatek";
}
```

WYJĄTKI Z WYKORZYSTANIEM KLAS

```
class Wyjatek{
    public:
    std::string _message = "Wystapil wyjatek klasy Wyjatek";
};
```

```
try{
    throw Wyjatek();
}
catch(Wyjatek& w){
    std::cout << w._message << std::endl;
}

return 0;
```

WYJĄTKI Z WYKORZYSTANIEM DZIEDZICZENIA

```
class Wyjatek{
    public:
    std::string _message = "Wystapil wyjatek klasy Wyjatek";
};

class PochodnyWyjatek: public Wyjatek{
    public:
    PochodnyWyjatek(){
        _message = "Wystapil wyjatek klasy PochodnyWyjatek";
    }
};
```

```
for(int i = 0; i < 2; i++){
    try{
        if(i == 0)
            throw Wyjatek();

        if(i == 1)
            throw PochodnyWyjatek();
    }
    catch(PochodnyWyjatek& w){
        std::cout << w._message << std::endl;
    }
    catch(Wyjatek& w){
        std::cout << w._message << std::endl;
    }
}
```

WYJĄTKI Z WYKORZYSTANIEM KONSTRUKTORA

```
class MyClass{
    public:
    MyClass(int i){
        if(i == 0)
            throw std::string("Zero jako parametr");
    }
};
```

WYJĄTKI W BIBLIOTECE STANDRDOWEJ

std::exception

- klasa bazowa dla standardowych wyjątków
- Wszystkie obiekty wyjątków wyrzucane przez komponenty ze standardowej biblioteki pochodza z tej klasy
- Wszystkie standardowe wyjątki mogą być wyłapywane przez ten typ referencji

It is declared as:

C++98 C++11

```
1 class exception {
2 public:
3   exception () noexcept;
4   exception (const exception&) noexcept;
5   exception& operator= (const exception&) noexcept;
6   virtual ~exception();
7   virtual const char* what() const noexcept;
8 }
```

https://cplusplus.com/reference/exception/exception/

WYJĄTKI W BIBLIOTECE STANDRDOWEJ

Typ	Znaczenie
std::logic_error	Ogólny błąd – naruszenie założenie, niezmiennik itp.
std::runtime_error	Ogólny błąd – bład czasu wykonania (możliwy do wykrycia tylko w czasie wykonania)
std::invalid_argument	Przekazano nieprawidłowy argument
std::domain_error	Błąd dziedziny operacji (matematycznej), np. dzielenie przez 0
std::length_error	Przekroczenie maksymalnego dozwolonego rozmiaru
std::out_of_range	Argument poza dopuszczalnym zakresem
std::range_error	Przekroczenie zakresu w obliczeniach, np. Wynik obliczeń jest poza zakresem docelowego typu
std::overflow_error	Przepełnienie liczby
std::underflow_error	Niedopełnienie liczby

ZADANIE 2

Korzystając z poprzedniego zadania, rozszerzmy je o dodatkowe kryteria.

- Wartości, które mogą przyjmować współrzędna to <0,20>. Inne wartości są nieakceptowalne. (obsłuż przy każdej operacji)
- Dodaj operator dzielenia przez scalar -> obsłuż dzielenie przez 0

ZADANIE 3

Napisz funckje, która będzie przeszukiwała tekst w poszukiwaniu słów niedozwolonych. Jeśli niedozowolone słowo będzie znajdowało się w tekście wyrzuć i obsłuż wyjątek.

ZADANIE 4

Przerób ostatni kod, tak aby powstała klasa bazowa Vec, która będzie zawierała wcześniejsze wykonane metody. Klasami pochodnymi będą Vec1, Vec2, Vec3. Współrzędne wektorów mają być przechowywane w std::vector.

Dodaj metodę at, która będzie zwracała odpowiednią współrzędną. Obsłuż możliwe błędy.

THANK YOU

Kamil Legierski

kamil.legierski@zeiss.com

www.zeiss.com