Instrukcja laboratoryjna nr 6

Programowanie w języku C 2

(C++ poziom zaawansowany)

Wyjątki (exceptions)

dr inż. Jacek Wilk-Jakubowski

mgr inż. Maciej Lasota

dr inż. Tomasz Kaczmarek

Wstęp

Wyjątki (ang. Exceptions) to mechanizm obsługi sytuacji wyjątkowych (zazwyczaj błędów). W trakcie działania programu zawsze może dojść do powstania sytuacji nietypowej trudnej do uniknięci. Można jednak przewidywać iż taka sytuacja wystąpi i się na nią odpowiednio przygotować. Tak jak Java, C# i wiele innych języków obiektowych, język C++ umożliwia obsługę wyjątków, czyli sytuacji, gdy powstaje w czasie wykonania programu *błąd* uniemożliwiający dalszy jego przebieg (na przykład dzielenie przez zero, wywołanie metody za pomocą pustego wskaźnika, błąd odczytu/zapisu strumienia). Normalnie, powstanie takiej sytuacji powoduje przerwanie programu, zwykle z mało czytelnym komunikatem o naturze błędu. Obsługa wyjątków pozwala na zdefiniowanie przez programistę akcji, które należy podjąć po powstaniu sytuacji wyjątkowej w szczególności programista może świadomie podjąć decyzję o ich całkowitym zignorowaniu.

W języku C++ wprowadzono mechanizm pozwalający programiście na reagowanie na sytuacje błędne czy nietypowe. Gdy taka sytuacja wystąpi, programista może wygenerować wyjątek. Wyjątek to **obiekt** pewnej klasy. Wygenerowanie wyjątku polega na przekazaniu obiektu opisującego wyjątek z fragmentu kodu, w którym wystąpił problem, do fragmentu, w którym przewidziano jego obsługę. Wygenerowanie (zgłoszenie) wyjątku powoduje przerwanie wykonywania sprawiającego problemy kodu i przejście do obsługi sytuacji problematycznej. Obsługa ta może znajdować się w innym miejscu kodu. Wyjątek jest obiektem, jego klasa określa typ sytuacji wyjątkowej. Obiekt może w sobie posiadać pola oraz funkcje składowe, pozwalające na sprecyzowanie informacji o zaistniałej sytuacji wyjątkowej.

Wyjątki w C++ możemy podzielić na:

- **Wyjątki własne**,
- **Wyjątki standardowe**.

Do obsługi wyjątków wykorzystuje się blok **try...catch**. Definicja bloku **try...catch** wygląda następująco:

```
try {
     // "Wyrzucenie" wyjątku:
     if(coś poszło nie tak)
          throw wyjątek;
     // KOD
}
catch (typ nazwa) {
     // Obsługa wyjątku
}
```

- **throw** – jest operatorem, po nim następuje wyrażenie które określi rodzaj wyjątku, mogą być różne typy/klasy wyjątków;
- **catch** – może pojawić się tylko po **try** lub po innym **catch**, w nawiasie określamy typ wyjątku i zmienną/obiekt która przekazuje informacje o wyjątku.

Jeżeli do obsługo błędu wystarczy sam fakt jego wykrycia to podajemy typ nie podając zmiennej/obiektu. Blok po catch nazywamy *procedurą obsługi wyjątku*. Procedura obsługi wyjątku może znajdować się w tej samej funkcji w której następuje zgłoszenie wyjątku.

Wyjątki wysłane w zakresie instrukcji **try**, ale nieprzechwycone przez odpowiedni blok **catch** zostają odebrane przez funkcję obsługi z parametrem *„wielokropek”* **catch(...)**. Taki blok łapie wszystkie błędy (pozostałe nie pasujące do innych bloków catch błędy).

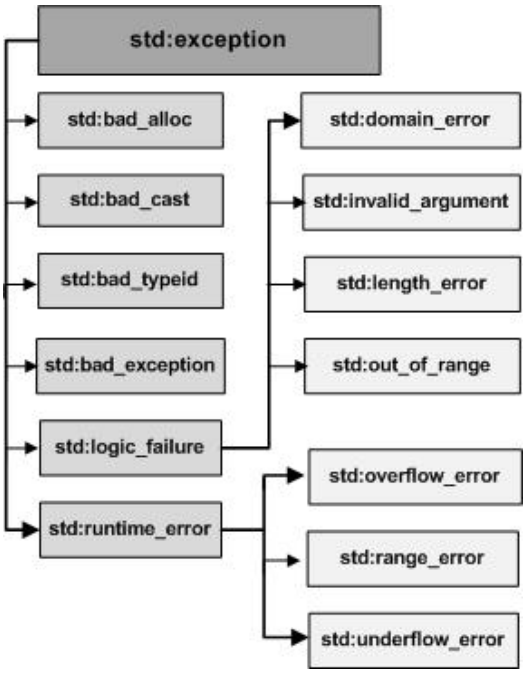
```
try {
     // KOD
}
catch (int ex) {
  cout << "wyjątek int";
}
catch (float ex) {
  cout << "wyjątek float";
}
catch (...) {
  cout << "wyjątek domyślny";
}
```

Przykłady implementacji i użycia bloku try...catch:

```
// Listing 1
try {
     int n;
     cout << "Podaj liczbę: ";
     cin >> n;

     // Czy ujemna?
     if (n < 0)     throw 1;
     // Czy zero?
     if (n == 0)    throw 2;
     // Czy za duża?
     if (n > 100000) throw 3;
```

```
        // Wszystko OK:
        cout << "Kwadrat liczby = " << n*n << endl;
    }
    catch (int x) {
        cout << "Wyjątek nr " << x << endl;
    }

    // Listing 2
    int FunkcjaWczytująca() {
        try {
            int A, B;
            cout << "Podaj dwie liczby: ";
            cin >> A >> B;

            if (A < 0)      throw "Liczba A ujemna!";
            if (B < 0)      throw "Liczba B ujemna!";
            if (B == 0)     throw "Dzielenie przez zero!";

            int wynik = A / B;

            if (wynik == 0)     throw "Wynik zerowy!";
            cout << "WYNIK = " << wynik << endl;
        }
        catch (char *s) {
            cout << "WYJĄTEK: " << s << endl;
        }
    }
```

W języku C++ można zdefiniować własne klasy wyjątków (tworząc własne klasy lub dziedzicząc po standardowej klasie **exception**), albo do opisu wyjątku wykorzystać już istniejące typy/klasy standardowe. W bibliotekach klas korzystających z wyjątków zazwyczaj korzysta się ze specjalnych klas do określania rodzaju wyjątków. Stosowanie klas jako typów wyjątków jest wygodne, możemy tworzyć klasy o nazwach opisujących typy wyjątków. Klasa standardowa **std::exception** to klasa bazowa wyjątków wykorzystywanych przez wiele klas standardowych. Najważniejsza metoda klasy to metoda **what()** która zwraca tekstowy opis wyjątku.



Przykłady implementacji i użycia własnej klasy wyjątku:

```
// Listing 3
#include <iostream>
#include <exception>

using namespace std;

class MojWyjatek: public exception {
  virtual const char* what() const throw() {
    return "Moj wyjatek";
  }
};

int main () {
  MojWyjatek myex;

  try  {
    throw myex;
  }
  catch (exception& e) {
    cout << e.what() << endl;
  }
  return 0;
}
```

Przykłady implementacji i użycia klasy standardowej wyjątku:

```
// Listing 4
#include <exception>
using namespace std;

try {
      int* myarray = new int[1000];
}
catch (exception& e) {
      cout << "Wyjątek: " << e.what() << endl;
}
```

Zadania do wykonania

1. Napisać program wyliczający pierwiastki równania kwadratowego w postaci:

$$ax^2 + bx + c = 0,$$

Program powinien pytać użytkownika o podanie trzech parametrów **a**, **b** oraz **c**. W programie należy wykorzystać obsługę wyjątków zabezpieczającą program przed nietypowymi sytuacjami jak: *dzielnie przez zero*, *pierwiastek z liczby ujemnej* itp. itd.

2. Napisać program, w którym zostaną zaimplementowane **własne** klasy wyjątków. Klasy te należy wykorzystać do obsługi wyjątków w programie wyliczającym następujący wzór matematyczny:

$$Wynik = \frac{\sqrt{A}}{B}$$

Należy zaprezentować działanie tak stworzonych klas dla różnych rodzajów danych wejściowych.

3. Napisać program, który będzie modyfikacją zadania nr 1 (laboratorium nr 2) implementującego stos przy pomocy kontenera list. Należy zabezpieczyć program tak aby obsługiwał wyjątki.

4. Napisać program, który będzie modyfikacją zadania nr 2 (laboratorium nr 2) implementującego listę uczniów w klasie przy pomocy kontenera set. Należy zabezpieczyć program tak aby obsługiwał wyjątki.