C++ - Wskaźnik this

Ogólne

Aplikacja prezentuje sposoby użycia wskaźnika *this* w programowaniu obiektowym. Wersja rozbudowana dodatkowo zawiera kontrolę danych wejściowych oraz kolorowanie składni.

Uwaga: Nigdy nie wolno kopiować kodu z PDF-a, gdyż zawiera on niewidoczne znaki końca linii i tzw. twarde odstępy. Kod należy przepisać ze zrozumieniem.

Kod - wersja podstawowa (*Windows & Linux*)

```
#include <iostream>

using namespace std;

class spisOsob {
private:
   string osoba;
   int wiek;
public:
   void uczniowie(string ziomek, int jakStaryJest) {
      cout << "Wynik działania funkcji 'uczniowie()':" << endl;
      cout << "Uczeń: " << ziomek << endl;
      cout << "Wiek: " << jakStaryJest << endl;
      this->osoba = ziomek;
      this->wiek = jakStaryJest;
   }
};


int main() {
   string name;
   int age;

   cout << "===== PROGRAMOWANIE OBIEKTOWE: THIS =====" << endl;

   // Na podstawie klasy "spisOsob" tworzę obiekt (instancję) o nazwie "obiektSpisOsob":
   spisOsob obiektSpisOsob;
   // Tego poniżej zrobić nie możemy, bo zmienna "osoba" jest prywatna:
   // obiektSpisOsob.osoba;
   // Ale możemy zrobić to:
   cout << "Podaj imię ucznia: ";
   cin >> name;
   cout << "Podaj wiek ucznia: ";
   cin >> age;
```

```
    obiektSpisOsob.uczniowie(name, age); // Przekazujemy argumenty

    cout << "Naciśnij ENTER, aby zakończyć..." << endl;
    cin.get();
    return 0;
}
```

Kod - wersja rozbudowana (*Windows*)

```
#include <iostream>
#include <windows.h> // Potrzebne do aktywacji kolorów w konsoli

using namespace std;

class spisOsob {
private:
    string osoba;
    int wiek;
public:
    // Poniżej, metoda (funkcja) posiadająca parametry.
    // Nazwy parametrów mogą być takie same jak zmienne prywatne (osoba, wiek), ale mogą
    // być też odmienne:
    void uczniowie1(string ziomek, int jakStaryJest) {
        cout << "\nWynik działania funkcji 'uczniowie1()':" << endl;
        cout << "Uczeń: \033[1;32;40m" << ziomek << "\033[0m" << endl;
        cout << "Wiek: \033[1;32;40m" << jakStaryJest << "\033[0m" << endl;
        // Do zmiennej prywatnej (tu mamy do niej dostęp) przypisujemy wartość zmiennej
        // globalnej (publicznej). Czyli zmienna "osoba" przejmie wartość wysłaną do zmiennej (do
        // parametru) "ziomek". W tym momencie zmienna prywatna staje się poniekąd zmienną
        // publiczną (bo jest zdefiniowana także w sekcji public). Będziemy mieli do niej dostęp w
        // funkcji main():
        this->osoba = ziomek;
        this->wiek = jakStaryJest;
        // W następnej funkcji / metodzie "uczniowie3()" możemy przetestować, czy te zmienne są
        // widoczne.
    }

    // Zakomentuj funkcję "uczniowie1()" lub "uczniowie2()", aby przetestować jedną z nich.
    void uczniowie2(string osoba, int wiek) {
        cout << "\n\nWynik działania funkcji 'uczniowie2()':" << endl;
        cout << "Uczeń: \033[1;32;40m" << osoba << "\033[0m" << endl;
        cout << "Wiek: \033[1;32;40m" << wiek << "\033[0m" << endl;
        // Taka sytuacja (gdy nazwy parametrów są takie same jak nazwy zmiennych prywatnych),
        // ma więcej sensu, gdy chcemy użyć "this", bo wskazujemy, o którą zmienną "osoba" nam
        // chodzi:
        this->osoba = osoba;
        this->wiek = wiek;
        // W następnej funkcji / metodzie "uczniowie3()" możemy przetestować, czy te zmienne są
```

```
        // widoczne.
    }

    void uczniowie3() {
        cout << "\n\nPoniżej wypisujemy zawartość prywatnych zmiennych po użyciu 'this':" <<
        endl;
        cout << "Uczeń: \033[1;32;40m" << osoba << "\033[0m" << endl;
        cout << "Wiek: \033[1;32;40m" << wiek << "\033[0m" << endl;
    }
};

// ============== CZY WPROWADZONO LICZBĘ? ===============
// Funkcja domyślnie zwraca prawdę, chyba że wykryje fałsz.
// Sprawdzany jest każdy znak z osobna (w pętli).
bool czyToJestLiczba(string liczbaJakoString) {
    for (int licznik = 0; licznik < liczbaJakoString.length(); licznik++) {
        if (isdigit(liczbaJakoString[licznik]) == false) {
        return false;
        }
    }
    return true;
}


int main() {
    // ====================== KOLOROWANIE NAPISÓW ==================
    // Tworzymy tzw. uchwyt do tego, co będzie pojawiać się na konsoli (do bufora konsoli):
    HANDLE konsola = GetStdHandle(STD_OUTPUT_HANDLE);
    // Aktywujemy virtualny terminal i to, co będzie się na nim pojawiać:
    #ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING // Jeśli nie jest zdefiniowany, to:
    #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
    #endif
    // Aby powyższe działało, musimy aktywować i to, co poniżej:
    #ifndef ENABLE_PROCESSED_OUTPUT // Jeśli nie jest zdefiniowany, to:
    #define ENABLE_PROCESSED_OUTPUT 0x0001
    #endif

    // Wartość trybu (intup lub output). Słowo "dw" to skrót od "Display Window",
    // jest to jednak nazwa zmiennej, i może być inna:
    DWORD dwMode = 0;
    dwMode |= ENABLE_PROCESSED_OUTPUT | ENABLE_VIRTUAL_TERMINAL_PROCESSING;
    SetConsoleMode(konsola, dwMode);
    // ================= KOLOROWANIE NAPISÓW - KONIEC ==============

    string name, wiekJakoString = "0";
    int wiekJakoInt = 0;
    bool wprowadzLiczbe = true; // W C++ zmienne logiczne nie mają domyślnej wartości
```

```
    cout << "\n\n\033[1;36;40m===== PROGRAMOWANIE OBIEKTOWE: THIS =====\033[0m\
    n\n" << endl;

    // Na podstawie klasy "spisOsob" tworzę obiekt (instancję) o nazwie "obiektSpisOsob":
    spisOsob obiektSpisOsob;
    // Tego poniżej zrobić nie możemy, bo zmienna "osoba" jest prywatna:
    // obiektSpisOsob.osoba;
    // Ale możemy zrobić to:
    cout << "Podaj imię ucznia: ";
    cin >> name;

    while (wprowadzLiczbe) {
        cout << "Podaj wiek ucznia: ";
        cin >> wiekJakoString;
        if (czyToJestLiczba(wiekJakoString) == false) {
            cout << "\n\n\033[1;31;40mTo nie jest liczba naturalna! Spróbuj jeszcze raz...\033[0m"
            << endl;
        }
         else {
            wprowadzLiczbe = false; // Jeśli wprowadzono liczbę, wyłącz pętlę WHILE
        }
    }


    wiekJakoInt = stoi(wiekJakoString);
    obiektSpisOsob.uczniowie1(name, wiekJakoInt); // Przekazujemy argumenty
    obiektSpisOsob.uczniowie2("Franek", 18);
    obiektSpisOsob.uczniowie3(); // Próbujemy wypisać zawartość zmiennych "osoba" i
    // "wiek"

    cout << "Naciśnij ENTER, aby zakończyć..." << endl;
    system("pause > nul");
    return 0;
}
```

Kod - wersja rozbudowana (*Linux*)

```
#include <iostream>

using namespace std;

class spisOsob {
private:
    string osoba;
    int wiek;
public:
```

```
    // Poniżej, metoda (funkcja) posiadająca parametry.
    // Nazwy parametrów mogą być takie same jak zmienne prywatne (osoba, wiek), ale mogą
    // być też odmienne:
    void uczniowie1(string ziomek, int jakStaryJest) {
        cout << "\nWynik działania funkcji 'uczniowie1()':" << endl;
        cout << "Uczeń: \033[1;32;40m" << ziomek << "\033[0m" << endl;
        cout << "Wiek: \033[1;32;40m" << jakStaryJest << "\033[0m" << endl;
        // Do zmiennej prywatnej (tu mamy do niej dostęp) przypisujemy wartość zmiennej
        // globalnej (publicznej). Czyli zmienna "osoba" przejmie wartość wysłaną do zmiennej (do
        // parametru) "ziomek". W tym momencie zmienna prywatna staje się poniekąd zmienną
        // publiczną (bo jest zdefiniowana także w sekcji public). Będziemy mieli do niej dostęp w
        // funkcji main():
        this->osoba = ziomek;
        this->wiek = jakStaryJest;
        // W następnej funkcji / metodzie "uczniowie3()" możemy przetestować, czy te zmienne są
        // widoczne.
    }

    // Zakomentuj funkcję "uczniowie1()" lub "uczniowie2()", aby przetestować jedną z nich.
    void uczniowie2(string osoba, int wiek) {
        cout << "\n\nWynik działania funkcji 'uczniowie2()':" << endl;
        cout << "Uczeń: \033[1;32;40m" << osoba << "\033[0m" << endl;
        cout << "Wiek: \033[1;32;40m" << wiek << "\033[0m" << endl;
        // Taka sytuacja (gdy nazwy parametrów są takie same jak nazwy zmiennych prywatnych),
        // ma więcej sensu, gdy chcemy użyć "this", bo wskazujemy, o którą zmienną "osoba" nam
        // chodzi:
        this->osoba = osoba;
        this->wiek = wiek;
        // W następnej funkcji / metodzie "uczniowie3()" możemy przetestować, czy te zmienne są
        // widoczne.
    }

    void uczniowie3() {
        cout << "\n\nPoniżej wypisujemy zawartość prywatnych zmiennych po użyciu 'this':" <<
        endl;
        cout << "Uczeń: \033[1;32;40m" << osoba << "\033[0m" << endl;
        cout << "Wiek: \033[1;32;40m" << wiek << "\033[0m" << endl;
    }
};

// ============== CZY WPROWADZONO LICZBĘ? ===============
// Funkcja domyślnie zwraca prawdę, chyba że wykryje fałsz.
// Sprawdzany jest każdy znak z osobna (w pętli).
bool czyToJestLiczba(string liczbaJakoString) {
    for (int licznik = 0; licznik < liczbaJakoString.length(); licznik++) {
        if (isdigit(liczbaJakoString[licznik]) == false) {
        return false;
        }
```

```
    }
    return true;
}


int main() {
    string name, wiekJakoString = "0";
    int wiekJakoInt = 0;
    bool wprowadzLiczbe = true; // W C++ zmienne logiczne nie mają domyślnej wartości

    cout << "\n\n\033[1;36;40m===== PROGRAMOWANIE OBIEKTOWE: THIS =====\033[0m\
    n\n" << endl;

    // Na podstawie klasy "spisOsob" tworzę obiekt (instancję) o nazwie "obiektSpisOsob":
    spisOsob obiektSpisOsob;
    // Tego poniżej zrobić nie możemy, bo zmienna "osoba" jest prywatna:
    // obiektSpisOsob.osoba;
    // Ale możemy zrobić to:
    cout << "Podaj imię ucznia: ";
    cin >> name;

    while (wprowadzLiczbe) {
        cout << "Podaj wiek ucznia: ";
        cin >> wiekJakoString;
        if (czyToJestLiczba(wiekJakoString) == false) {
            cout << "\n\n\033[1;31;40mTo nie jest liczba naturalna! Spróbuj jeszcze raz...\033[0m"
            << endl;
        }
        else {
            wprowadzLiczbe = false; // Jeśli wprowadzono liczbę, wyłącz pętlę WHILE
        }
    }

    wiekJakoInt = stoi(wiekJakoString);
    obiektSpisOsob.uczniowie1(name, wiekJakoInt); // Przekazujemy argumenty
    obiektSpisOsob.uczniowie2("Franek", 18);
    obiektSpisOsob.uczniowie3(); // Próbujemy wypisać zawartość zmiennych "osoba" i
    // "wiek"

    cout << "Naciśnij ENTER, aby zakończyć..." << endl;
    cin.get();
    return 0;
}
```