Laboratorium 0: „Środowisko Code::Blocks”

dr inż. Arkadiusz Chrobot

28 września 2015

1. Wprowadzenie

W tej instrukcji został zawarty krótki opis zintegrowanego środowiska do tworzenia programów w języku C, o nazwie Code::Blocks. Program ten wydawany jest na wolnej licencji GPL. Można go pobrać ze strony projektu http://www.codeblocks.org. Środowisko dostępne jest dla wszystkich popularnych systemów operacyjnych i współpracuje z wieloma kompilatorami języka C. Na zajęciach laboratoryjnych będziemy używać kompilatora `gcc`.

2. Konfiguracja

Po uruchomieniu środowiska należy zamknąć okno z poradą dnia, a następnie z górnego menu wybrać opcje `Settings` → `Compiler…`. W sekcji `Warnings` należy zaznaczyć pole "Enable all common compiler warnings (overrides many other settings) [-Wall]" i kliknąć przycisk OK. Wybrana opcja będzie zaznaczona po ponownym uruchomieniu środowiska. Nakazuje ona kompilatorowi dokładniejsze sprawdzanie kompilowanego programu pod względem zgodności ze składnią. Jeśli kompilator wykryje jakieś rozbieżności, to wygeneruje ostrzeżenie (ang. *warning*). Ostrzeżenie nie jest błędem, ale informacją od kompilatora, że pewien fragment kodu źródłowego jest zapisany wieloznacznie i w związku z tym musiał on podczas jego tłumaczenia przyjąć pewne założenia, które nie muszą zgadzać się z zamiarami programisty. W takiej sytuacji program wynikowy może działać inaczej niż zakładał jego twórca. Błąd w kodzie źródłowym (ang. *error*) powoduje natychmiastowe przerwanie kompilacji i plik z programem wykonywalnym (kod wynikowy) nie powstaje. Aby program działał poprawnie zawsze należy wyeliminować wszystkie błędy w nim zawarte oraz dążyć do poprawienia fragmentów generujących ostrzeżenia.

Aby ułatwić programiście znajdowanie błędów, środowisko Code::Blocks współpracuje z programem nazywanym debuggerem. Aby w pełni wykorzystać jego możliwości należy kompilatorowi nakazać umieszczenie dodatkowych informacji w kodzie wynikowym kompilowanego programu. Aby to zrobić musimy postępować tak jak wyżej, ale w wyświetlonym oknie dialogowym należy zaznaczyć opcję "Produce debugging symbols [-g]".

3. Tworzenie nowego projektu

Programy w środowisku Code::Blocks tworzone są w ramach projektów. Nowy projekt można utworzyć na kilka sposobów. Najprostszy to kliknięcie łącza "Create a new project" w głównym oknie aplikacji. Można również w górnym menu wybrać następujący ciąg opcji `Plik` → `New` → `Project…`. W oknie dialogowym, które się ukarze należy kliknąć ikonę "Console application", następnie wskazać język C (nie C++) i kliknąć przycisk „Dalej". W nowym formularzu należy podać nazwę projektu. Ponieważ na jej podstawie tworzone są nazwy plików, to nie zaleca się stosowania w nich spacji. Zamiast nich można użyć znaku podkreślenia (_). Po wprowadzeniu nazwy należy ponownie kliknąć przycisk „Dalej", a następnie „Zakończ". Na liście rozwijanej w oknie po prawej stronie pojawi się ikona z nazwą projektu, a poniżej niej ikona z katalogiem o nazwie "Sources". Po jej kliknięciu pojawi się ikona z plikiem o nazwie `main.c`. Pliki z rozszerzeniem `.c` zawierają kod źródłowy programu zapisany w języku C. Nazwa pliku może być dowolna, ale środowisko Code::Blocks zwyczajowo nazywa taki plik `main`. Po kliknięciu na tę nazwę w oknie głównym środowiska, czyli oknie edytora pojawi się zawartość tego pliku - program typu "Hello World!". Ma on trochę inną postać niż ta, którą podano na wykładzie. Do wypisywania komunikatu użyto w nim funkcji `printf()` zamiast `puts()`, stąd konieczność dodania na końcu komunikatu znaku nowego wiersza (`\n`), aby po wypisaniu komunikatu kursor przeszedł do kolejnego wiersza na ekranie. Nadmiarowo włączono w tym programie plik nagłówkowy o nazwie `stdlib.h`. Jeśli usuniemy instrukcję `#include<stdlib.h>` program nadal będzie działał i kompilował się bez ostrzeżeń. Proszę zwrócić uwagę, że wiersze kodu źródłowego w głównym oknie, które jest jednocześnie oknem edycji kodu źródłowego, są ponumerowane. Te numery nie są częścią kodu źródłowego. Są one dodawane przez środowisko celem ułatwienia pracy programistom.

4. Kompilacja i uruchomienie progamu

Kompilację programu w środowisku Code::Blocks można uruchomić na kilka sposobów. Najprostszym jest naciśnięcie kombinacji klawiszy `Ctrl+F9`. Proszę zwrócić uwagę, że dolne okno automatycznie przełącza się na zakładkę o tytule „Build log". Zawiera ona komunikaty kompilatora. Jeśli pojawiłby się tam informacje o błędach lub ostrzeżeniach, to kliknięcie w nie przenosi kursor w oknie edycji kodu źródłowego w miejsce gdzie przyczyna danego błędu lub ostrzeżenia (prawdopodobnie) występuje[1]. Dodatkowo te miejsca są odpowiednio oznaczone w oknie edycji kodu źródłowego. Skompilowany program można uruchomić przy pomocy kombinacji klawiszy `Ctrl+F10`. Można również wykonać obie czynności, tj. kompilację i uruchomienie programu za jednym razem naciskając klawisz `F9`. Więcej opcji związanych z kompilacją i uruchamianiem programu można znaleźć w pozycji „Build" górnego menu środowiska. Dostępne są również odpowiednie paski narzędzi ułatwiające wykonanie obu czynności za pomocą myszy. Zalecam przede wszystkim opanowanie skrótów klawiszowych. Jest najbardziej ergonomiczny i ekonomiczny sposób pracy ze środowiskiem programistycznym.

5. Debuggowanie programu

Środowisko Code::Blocks umożliwia krokowe (tj. wiersz po wierszu) wykonanie programu i obserwację wartości zmiennych. Taki sposób wykonania programu jest przydatny kiedy próbujemy ustalić miejsce, w którym występuje błąd. Aby uruchomić ten tryb pracy możemy postąpić dwojako. Pierwszy sposób polega na umieszczeniu kursora w miejscu od którego chcemy śledzić wykonanie programu i naciśnięciu klawisza `F4`. Program wykona się do tego miejsca i zatrzyma. Inny sposób polega na użyciu klawisza `F5` zamiast `F4`. Ustawia on w miejscu umieszczenia kursora pułapkę, co oznacza, że program będzie się zatrzymywał zawsze w tym miejscu do czasu zdjęcia pułapki. Do tej ostatniej czynności można ponownie użyć klawisza `F5`. Jeśli w programie jest więcej zdefiniowanych pułapek to można je usunąć wybierając z menu górnego następujące opcje `Debug` → `Remove all breakpoints`. Aby rozpocząć wykonanie programu z pułapkami należy nacisnąć klawisz `F8`. Po zatrzymaniu wykonania programu możemy wykonywać kolejne instrukcje naciskając sukcesywnie klawisz `F7`. Jeśli chcemy zbadać działanie funkcji zdefiniowanej w programie[2] to musimy użyć następującej opcji z menu górnego: `Debug` → `Step into`[3]. Aby zbadać jak zmieniają się wartości zmiennych należy otworzyć okno śledzenia za pomocą klikając następujące opcje górnego menu: `Debug` → `Debugging windows` → `Watches`. To okno można osadzić w dolnej części środowiska. Aby śledzić zmianę wartości konkretnej zmiennej należy wpisać jej nazwę w pierwszej kolumnie tabeli w oknie `Watches` i nacisnąć klawisz `Enter`. W ostatniej kolumnie tabeli pojawi się typ tej zmiennej (jeśli ona istnieje i jest widoczna w tej części programu, która bieżąco jest wykonywana), a w środkowej jej obecna wartość, która będzie się zmieniała, w trakcie krokowego wykonania programu.

6. Przykładowy program

Listing 1 zawiera kod źródłowy programu, w którym zdefiniowano funkcję obliczającą pole koła o nazwie `calculate_area()`. Ta funkcja ma jeden parametr wejściowy typu `double`, przez który przekazywana jest długość promienia, a zwraca wartość pola. Typ wartości zwracanej i parametru określa, że obie te liczby będą liczbami rzeczywistymi[4]. Funkcja ta jest wywoływana w głównej funkcji programu o nazwie `main()`. W tej funkcji program pyta użytkownika o długość promienia, następnie przy pomocy funkcji `scanf()` zapisuje ją w zmiennej `radius`. Jeśli długość promienia jest nieujemna to wartość tej zmiennej jest przekazywana do funkcji `calculate_area()`, która zwraca obliczoną wielkość pola do wywołania funkcji `printf()`. Ta ostania wypisuje na ekranie komunikat, wielkość promienia koła oraz jego pole. Jeśli promień miałby „ujemną długość", to program wypisze komunikat o błędzie. Te dwa przypadku są rozróżniane dzięki zastosowaniu instrukcji warunkowej `if`. **Uwaga!** Tak program powi-

[1] Zawsze należy sprawdzić oprócz wiersza, w którym jest kursor również wiersze sąsiadujące.

[2] Funkcja to wydzielony fragment programu, nazywany także podprogramem.

[3] Istnieje skrót klawiszowy dla tej opcji: `Shift+F7`, ale nie we wszystkich wersjach środowiska działa on poprawnie.

[4] Dokładniej są to liczby zmiennoprzecinkowe, które stanowią dosyć dobrą reprezentację liczb rzeczywistych w pamięci komputera.

nien działać, ale w jego kodzie celowo wprowadzono dwa błędy. Jeden jest błędem składni i ujawnia się podczas kompilacji, a drugi w trakcie wykonania programu.

```
#include <stdio.h>

#define PI 3.1415

double calculate_area(double r)
{
        return PI*r*r;
}

double radius;

int main()
{
        puts("Podaj długość promienia koła");
        scanf("%lf",&radius);
        //Program sprawdza, czy promień ma wartość większą lub równą zero.
        if(radius<=0.0)
                printf("Pole koła o promieniu %lf cm wynosi %lf cm^2.\n"
                        ,radius,calculate_area(radius))
        else
                puts("Podana długość promienia jest nieprawidłowa.");
        return 0;
}
```

Listing 1: Kod źródłowy programu obliczającego pole koła o zadanym promieniu.

7. Zadania

1. Zapoznaj się z treścią tej instrukcji. Naucz się podstawowej obsługi środowiska Code::Blocks.
2. Popraw błąd kompilacji w programie z listingu 1.
3. Znajdź i usuń błąd, który pojawia się w trakcie wykonania programu. Użyj do tego techniki debuggowania.
4. Sprawdź jakie inne opcje ułatwiające programowanie oferuje środowisko Code::Blocks. Postaraj się nauczyć niektórych z nich.