

# LABORATORIUM 1. WPROWADZENIE DO PROGRAMOWANIA STRUKTURALNEGO W JĘZYKU C. WPROWADZENIE DO ŚRODOWISKA PROGRAMISTYCZNEGO.

## Cel laboratorium:

Zaznajomienie ze strukturą prostego programu w języku C. Zaznajomienie z procesem pisania i uruchamiania programów w zintegrowanych środowiskach programistycznych Dev-C++ i Code::Blocks. Nabycie praktycznych umiejętności pracy w środowiskach: Dev-C++ i Code::Blocks.

## Zakres tematyczny zajęć:

- struktura programu w języku C,
- elementy środowiska programistycznego (Dev-C++ lub Code::Blocks),
- sposób tworzenia i zapisu nowego projektu,
- tworzenie kodu źródłowego programu, kompilacja i wykonanie,
- procesy edycji kodu źródłowego, kompilacji, linkowania, uruchomienia programu,
- błędy kompilacji,
- plik źródłowy a plik wykonywalny.

## Kompendium wiedzy:

### *Program w języku C – zasady:*

- Program w języku C jest zbiorem modułów.
- Każdy moduł składa się z globalnych deklaracji typów, zmiennych, funkcji.
- Dokładnie jeden moduł zawiera definicję funkcji głównej (main).
- Wykonanie programu polega na opracowaniu jego globalnych deklaracji, a potem na wykonaniu instrukcji funkcji main.
- Zakończenie programu następuje po wykonaniu instrukcji powrotu (return) w funkcji main lub po powrocie z funkcji exit.
- Instrukcje przeznaczone dla kompilatora (dyrektywy preprocesora) rozpoczynają się # i nie kończą średnikiem.
- Wielkość liter w identyfikatorach (nazwach własnych) ma znaczenie.
- Komentarze:

*/\*komentarz w C \*/*

*lub //komentarz w C i C++*

### *Struktura programu w C*

```
//dyrektywy preprocesora
int main() //nagłówek funkcji głównej
{
//deklaracje
//instrukcje
return 0; //instrukcja powrotu
}
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



**Proces programowania:**

1. Określenie celów programu.
2. Zaprojektowanie algorytmu programu.
3. Utworzenie projektu i edycja kodu źródłowego programu.

**Projekt** w środowiskach programistycznych - zbiór modułów kodu źródłowego i innych plików, które po kompilacji i łączeniu stają się pojedynczym plikiem EXE, czyli programem (zalecany oddzielny katalog).

4. Kompilacja i łączenie.

**Kompilacja** – tłumaczenie kodu źródłowego (\*.c) na kod maszynowy (\*.obj).

**Łączenie** – konsolidacja skompilowanych plików z bibliotekami do pliku wykonywalnego (\*.exe).

5. Uruchomienie programu.
6. Testowanie i usuwanie błędów.
7. Pielęgnowanie i modyfikacja programu.

**Pytania kontrolne:**

1. Objaśnij strukturę prostego programu w języku C.
2. Jak utworzyć nowy program w środowisku programistycznym Dev-C++ (Code::Blocks)?
3. Jak uruchomić program?
4. Na czym polega proces kompilacji?
5. Na czym polega proces łączenia?

**Zadania do analizy**

**Zadanie 1.1. Struktura prostego programu w języku C**

Przeanalizuj kod źródłowy prostego programu w języku C. Zwróć uwagę na komentarze.

```
#include <stdio.h>           /*dyrektywy kompilatora*/
#include <stdlib.h>          /*dyrektywy kompilatora*/

int main(int argc, char *argv[]) {    /*funkcja glowna*/
    int ocena;                      /*definicja zmiennej o nazwie ocena*/
    ocena=5;                        /*przypisanie zmiennej wartości 5 */
    /*wyswietlenie na ekranie*/
    printf("Zaczynam programowac w jezyku C\n");
    printf("Z przedmiotu \"Programowanie strukturalne\"
           otrzymam ocene %d\n",ocena);
    return 0;
}
```



## Zadanie 1.2. Zintegrowane środowiska programistyczne Dev-C++ i Code::Blocks

Przeanalizuj sposób pracy w zintegrowanym środowisku programistycznym **Dev-C++**:

- Pakiet instalacyjny Dev-C++: <http://www.bloodshed.net/devcpp.html>.
- Tworzenie nowego projektu: *Plik/ Nowy projekt/ Console Application* z opcją *Projekt C*.
- Zapis plików projektu: *Plik/Zapisz (Zapisz jako/Zapisz projekt jako /Zapisz wszystko)*.
- Otwarcie projektu/pliku: *Plik/Otwórz projekt lub plik*.
- **Uruchomienie programu:**
  - *Uruchom/Kompiluj* - kompilacja kodu aktualnego pliku źródłowego.
  - *Uruchom/ Uruchom* - uruchomienie programu.
  - *Uruchom/ Kompiluj i uruchom* - kompilacja przyrostowa plików źródłowych i uruchomienie.
  - *Uruchom/ Przebuduj wszystko* - skompilowanie od zera wszystkich plików źródłowych.

Przeanalizuj sposób pracy w zintegrowanym środowisku programistycznym **Code::Blocks**:

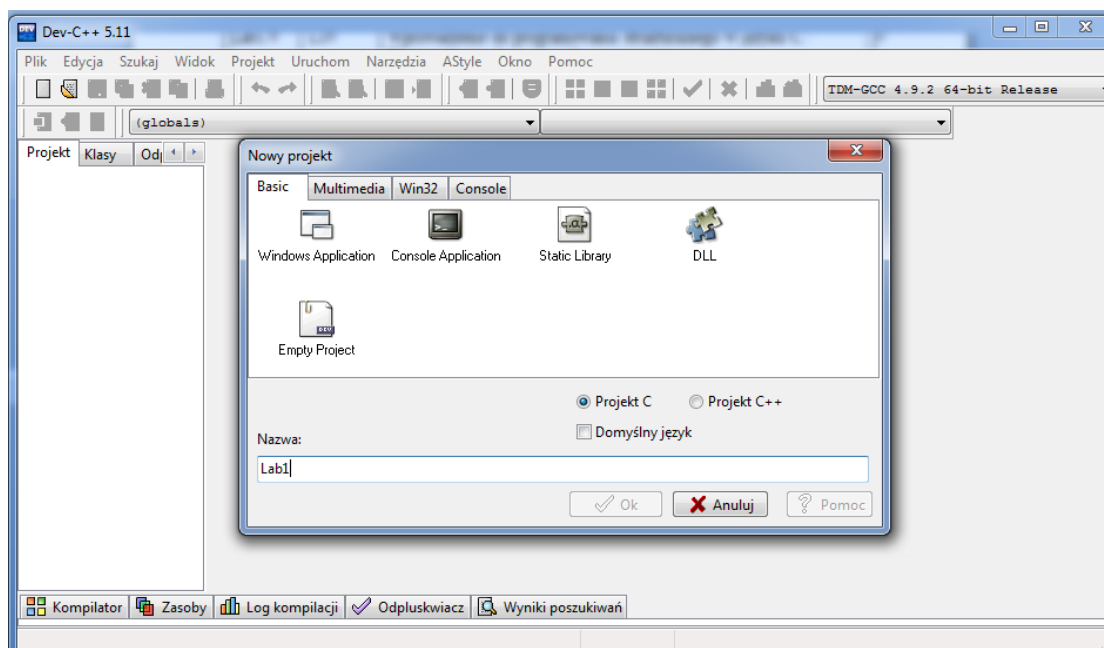
- Pakiet instalacyjny Code::Blocks: [www.codeblocks.org](http://www.codeblocks.org).
- Tworzenie nowego projektu: *File/New/Project/Console Application* z opcją *C*.
- Zapis plików projektu: *File/Save file (Save file as /Save Project as /Save all files)*.
- Otwarcie projektu/pliku: *File/Open*.
- **Uruchomienie programu:**
  - *Build/Build* - kompilacja kodu aktualnego pliku źródłowego.
  - *Build/Run* - uruchomienie programu.
  - *Build/Build and run* - kompilacja przyrostowa plików źródłowych i uruchomienie.
  - *Build/Rebuild* - skompilowanie od zera wszystkich plików źródłowych.

### Zadania do wykonania

## Zadanie 1.3. Uruchomienie prostego programu w zintegrowanym środowisku programistycznym Dev-C++ i testowanie błędów

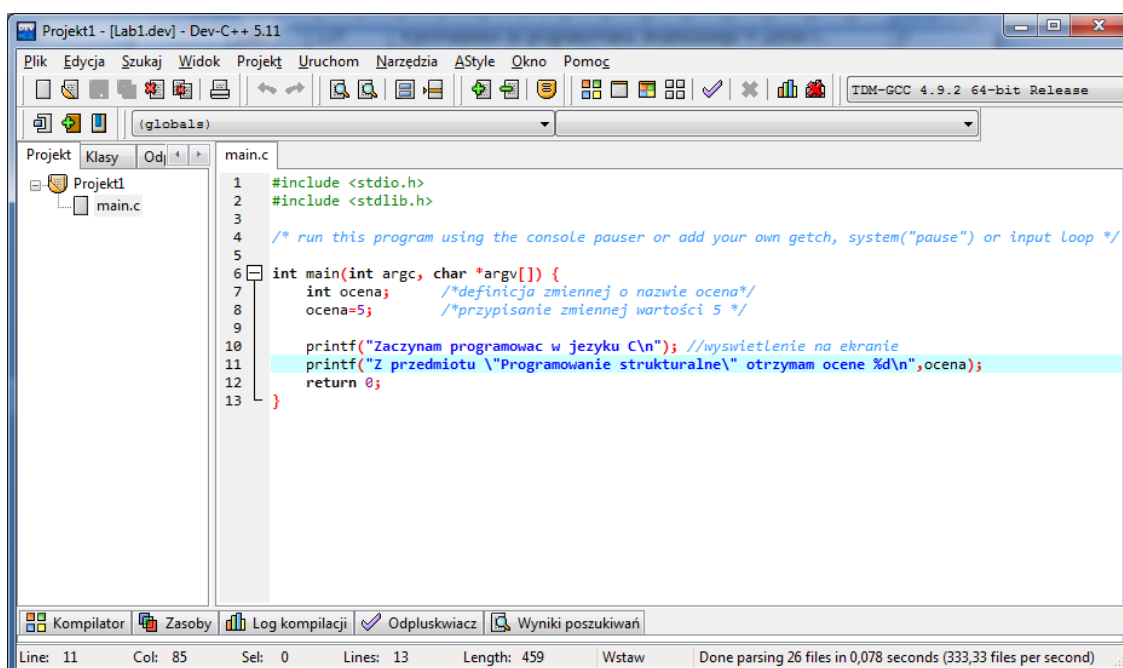
- Uruchom środowisko programistyczne Dev-C++.
- Utwórz nowy projekt w C jako aplikację konsolową (Rys.1.1).





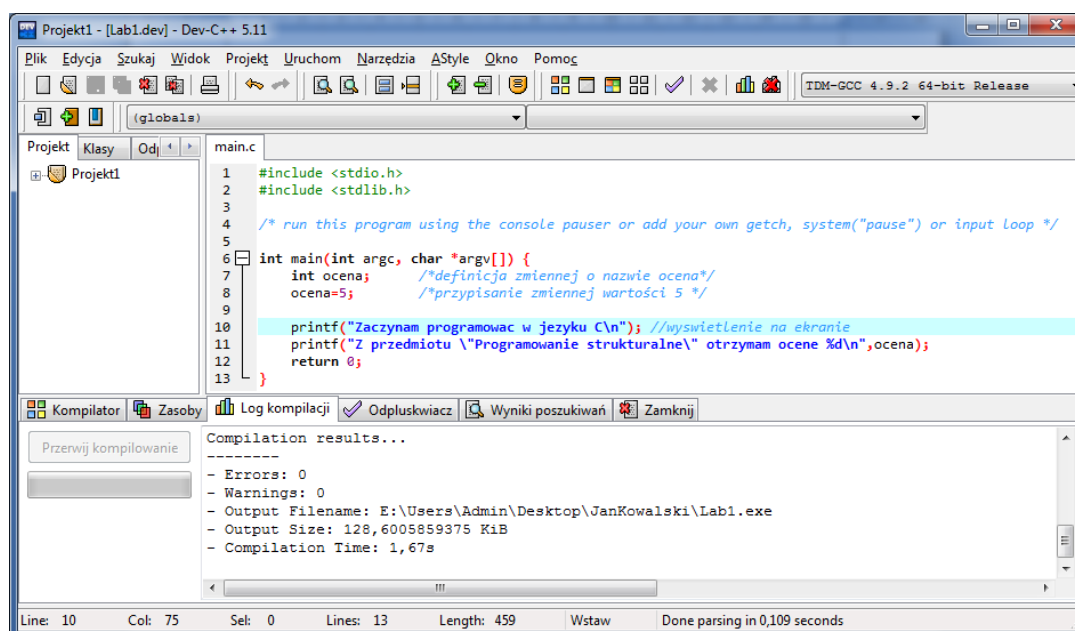
*Rys. 1. 1 Tworzenie projektu aplikacji konsolowej w Dev-C++*

- Zapisz plik pod nazwą Lab1 w katalogu o swoim nazwisku na pulpicie.
- Zapoznaj się z paskami narzędzi i menu środowiska programistycznego Dev-C++.
- Do edytora kodu w pliku main.c wprowadź kod źródłowy programu z zadania 1.1 (Rys. 1.2).



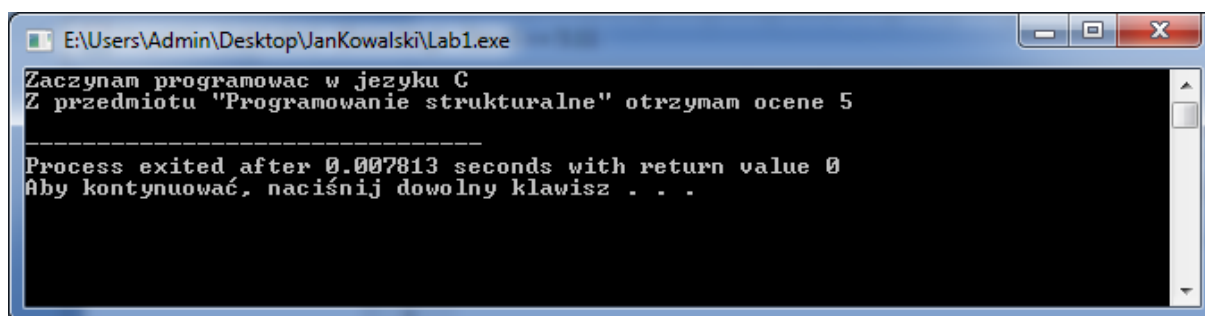
*Rys. 1. 2 Kod źródłowy programu z zadania 1.1 w Dev-C++*

- Wykonaj kompilację programu poleceniem *Uruchom/Kompiluj* i sprawdź jej poprawność (Rys. 1.3).



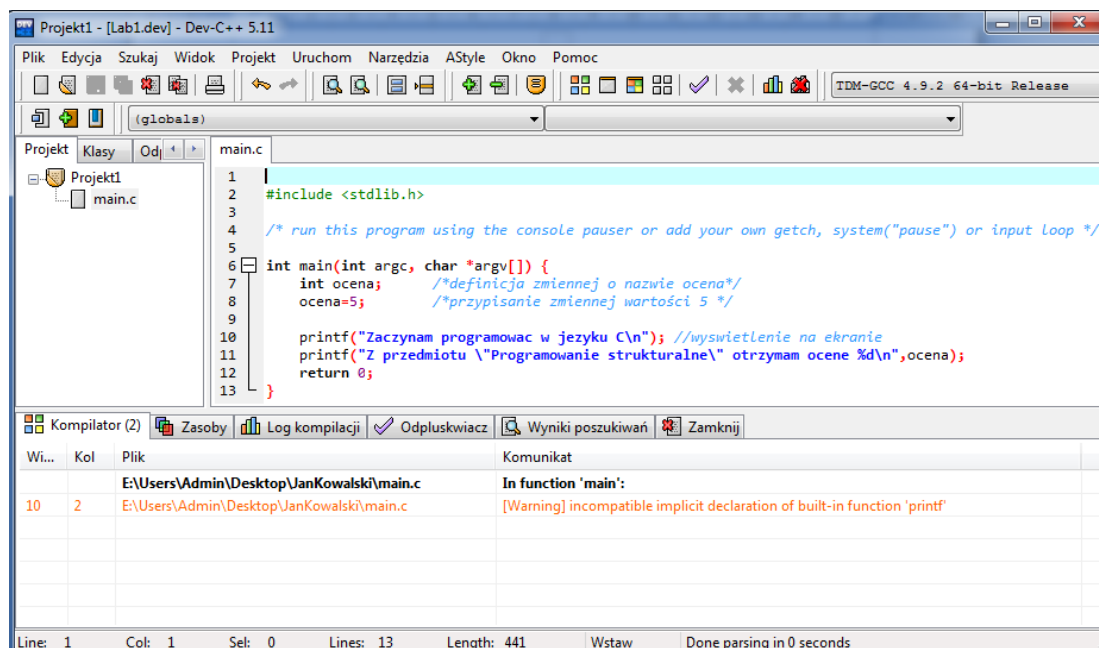
*Rys. 1. 3 Informacja o wyniku kompilacji programu*

- Uruchom program (*Uruchom*/*Uruchom*) i przeanalizuj jego rezultat (Rys. 1.4).



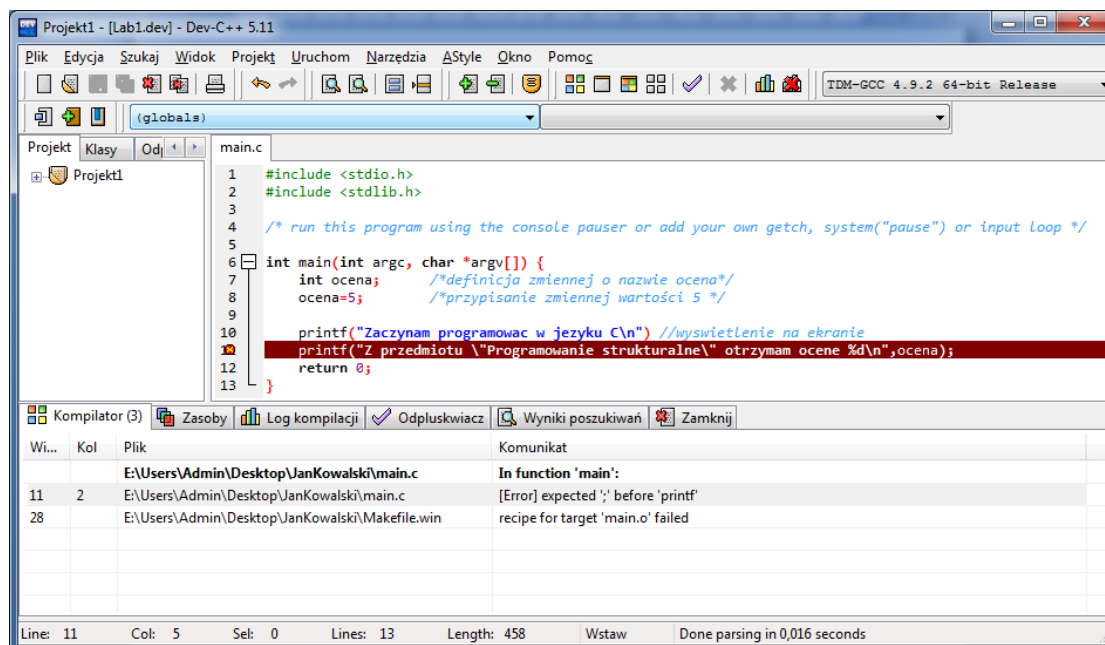
*Rys. 1. 4 Wynik uruchomienia programu*

- Wprowadź błąd w kodzie np. usuń wiersz 1 (brak deklaracji pliku nagłówkowego `stdio.h`) i ponownie skompiluj program. Przeanalizuj informację o rodzaju błędu (Rys.1.5).



Rys. 1. 5 Błąd łączenia

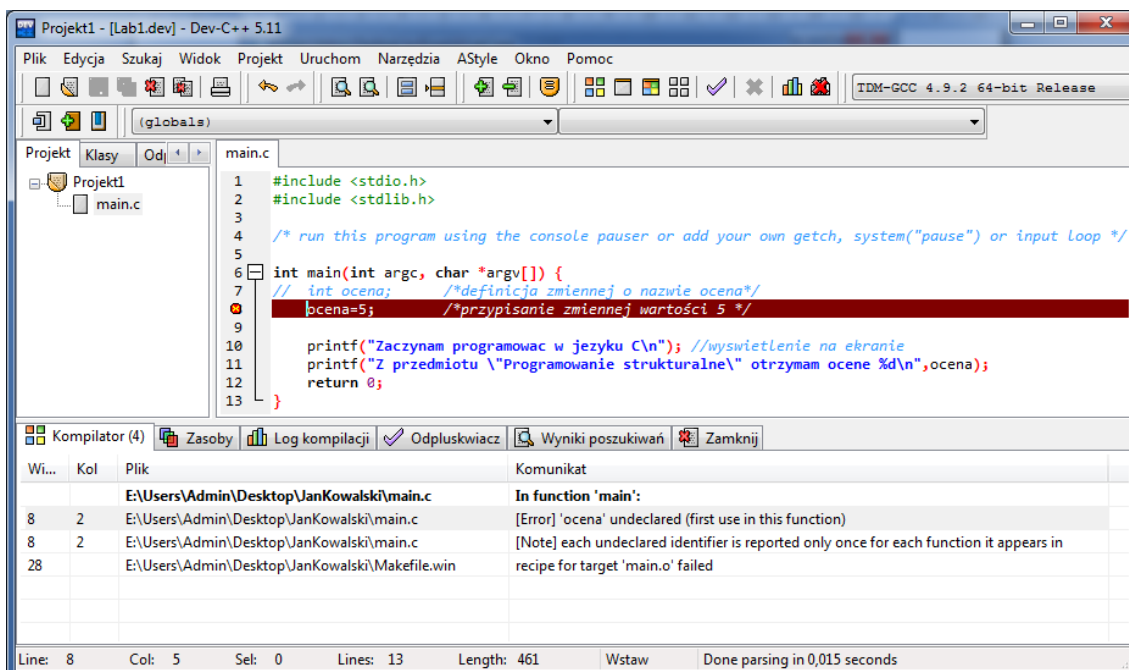
- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Uruchom/ Kompiluj i uruchom*.
- Wprowadź kolejny błąd w kodzie np. usuń średnik na końcu polecenia w wierszu 10 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędzie (Rys.1.6).



Rys. 1. 6. Błąd składni

- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Uruchom/ Kompiluj i uruchom*.

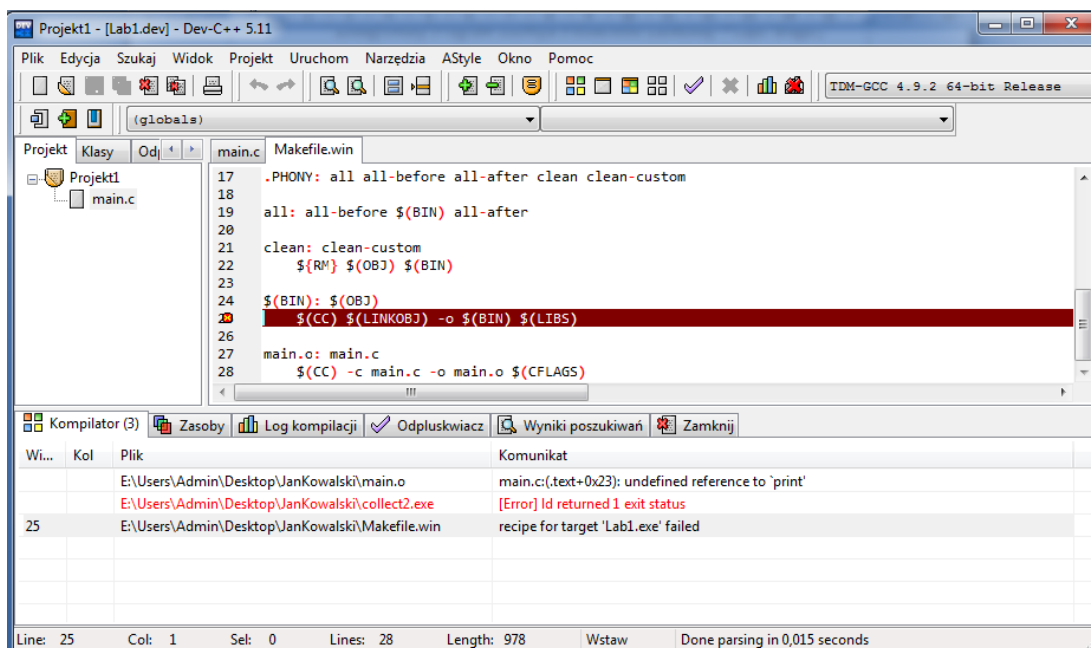
- Wprowadź kolejny błąd w kodzie np. zakomentuj wiersz 7 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.7).



Rys. 1. 7 Błąd semantyczny

- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Uruchom/ Kompiluj i uruchom*.
- Znajdź odpowiednie ikony realizowanych poleceń menu *Uruchom* na pasku narzędziowym.
- Wprowadź kolejny błąd w kodzie np. w wierszu 10 usuń literę f z funkcji printf (zła nazwa funkcji) i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.8).



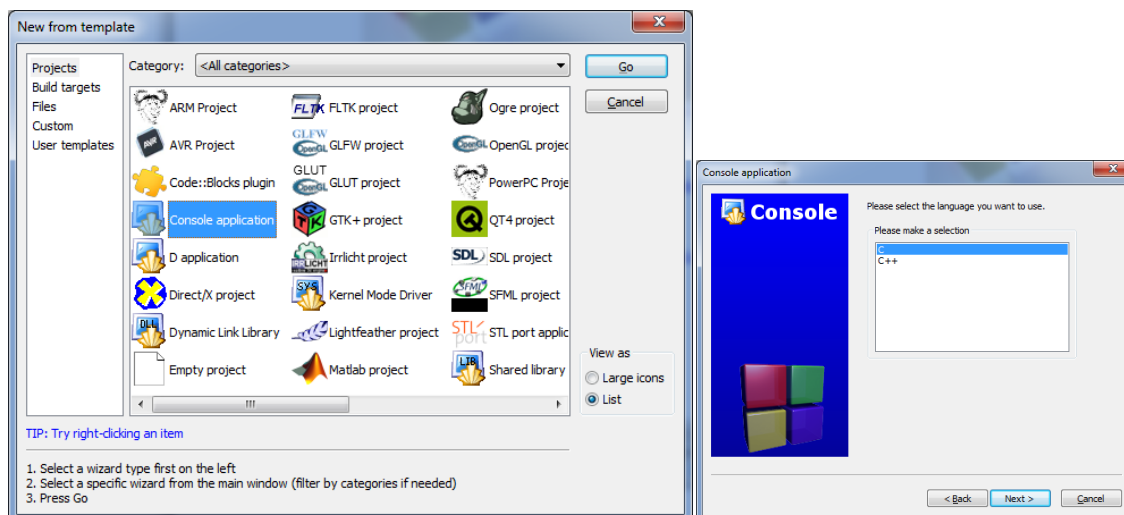


Rys. 1. 8 Błąd łączenia

- Popraw błąd w kodzie źródłowym i ponownie uruchom program przyciskiem .

#### Zadanie 1.4. Uruchomienie prostego programu w zintegrowanym środowisku programistycznym Code::Blocks

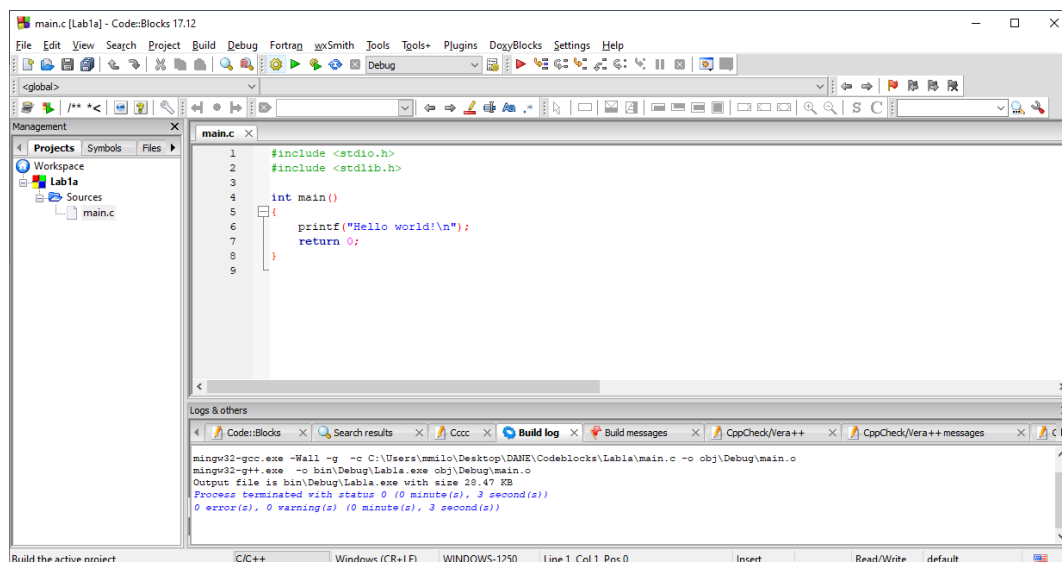
- Uruchom środowisko programistyczne Code::Blocks.
- Utwórz nowy projekt w C jako aplikację konsolową (Rys.1.9).



Rys. 1. 9 Tworzenie projektu aplikacji konsolowej w Code::Blocks

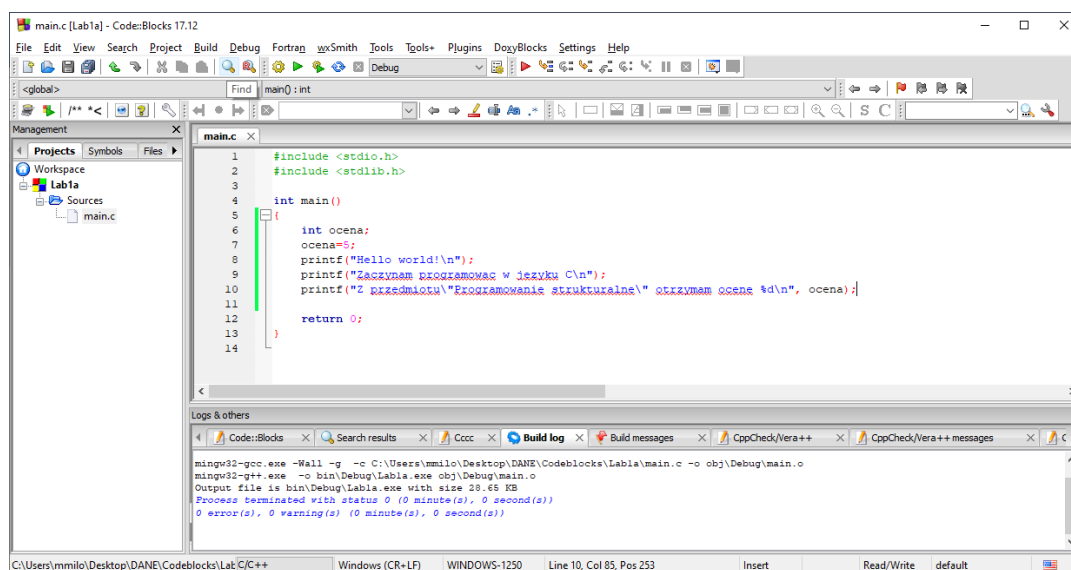
- Zapisz plik pod nazwą Lab1a w katalogu o swoim imieniu na pulpicie.
- Zapoznaj się z paskami narzędzi i menu środowiska programistycznego Code::Blocks (Rys.1.10).





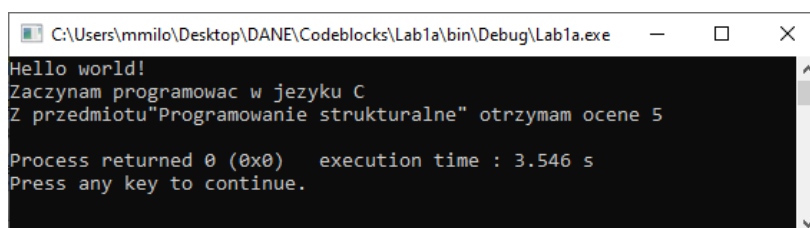
Rys. 1. 10 Nowy projekt w Code::Blocks

- Do edytora kodu w pliku `main.c` wprowadź kod źródłowy programu z zadania 1.1.
- Wykonaj kompilację programu poleceniem *Build/Build* i sprawdź jej poprawność (Rys. 1.11).



Rys. 1. 11 Kompilacja programu

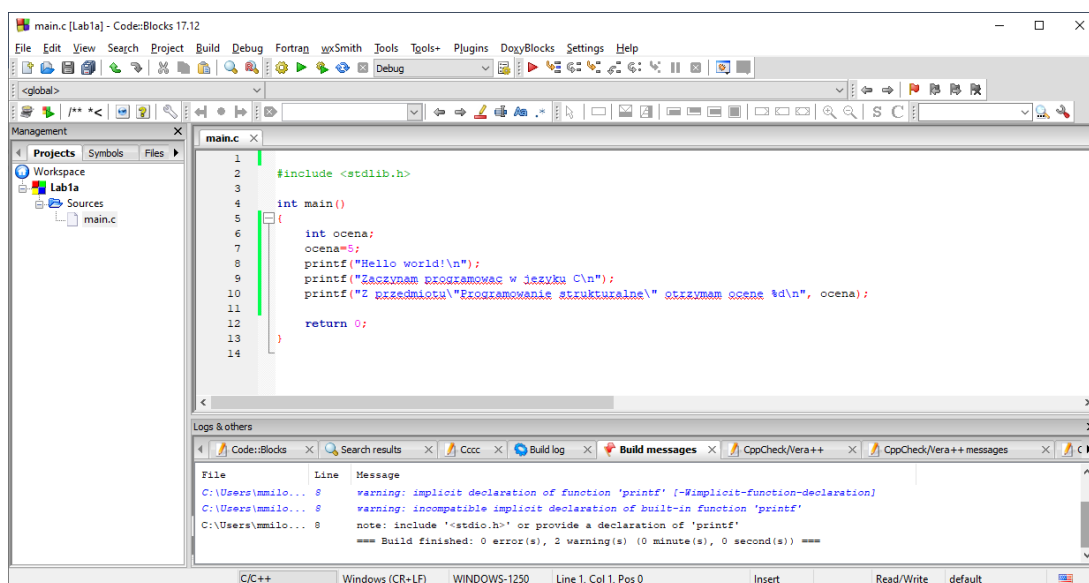
- Uruchom program i przeanalizuj jego rezultat (Rys.1.12).



Rys. 1. 12 Rezultat wykonania programu

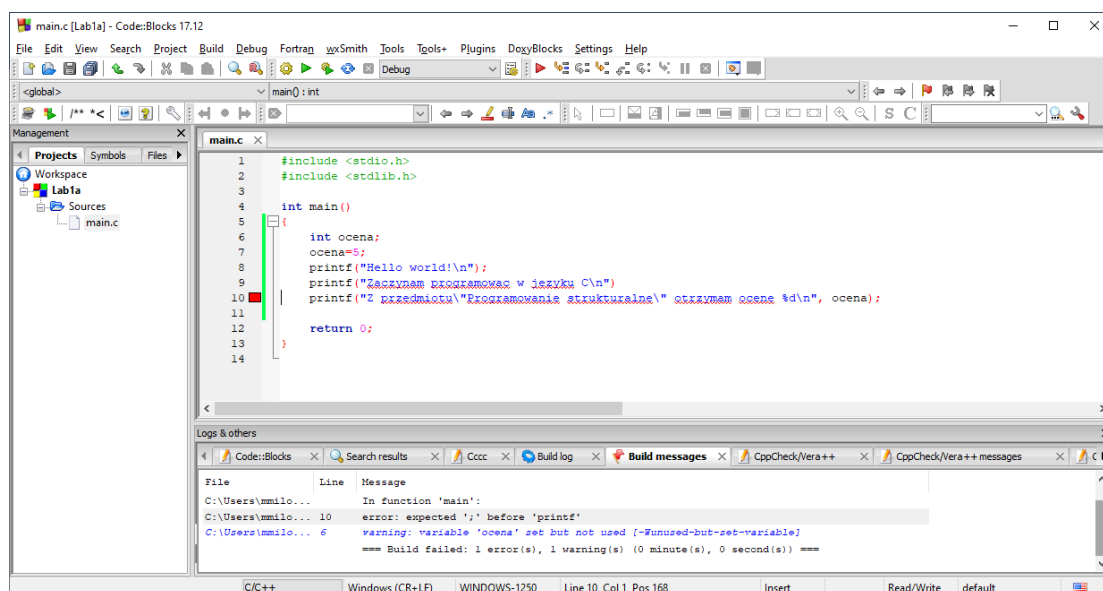


- Wprowadź celowo błąd w kodzie np. usuń wiersz 1 (brak deklaracji pliku nagłówkowego `stdio.h`) i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.13).



Rys. 1. 13. Błąd łączenia

- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Build/Build and run*.
- Wprowadź kolejny błąd w kodzie: usuń średnik na końcu polecenia w wierszu 9 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.14).

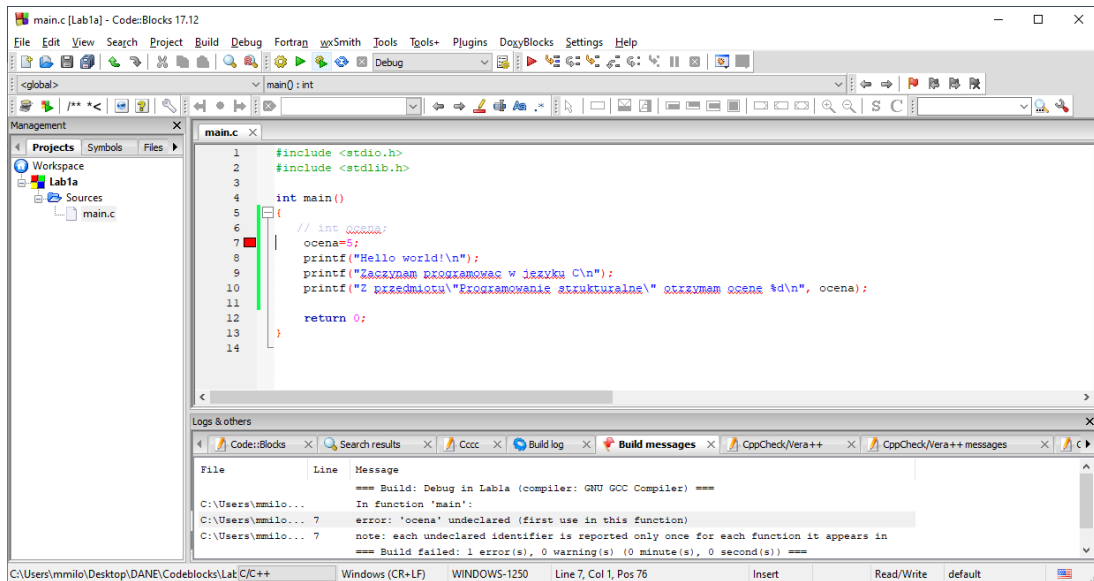


Rys. 1. 14 Błąd składni


- Popraw błąd w kodzie źródłowym i ponownie uruchom program poleceniem *Build/Build and run*.



- Wprowadź kolejny błąd w kodzie np. zakomentuj wiersz 6 i ponownie skompiluj program. Przeanalizuj informację kompilatora o rodzaju błędu (Rys.1.15).



Rys. 1. 15 Błąd semantyczny

- Znajdź odpowiednie ikony realizowanych poleceń menu *Build* na pasku narzędziowym.
- Popraw błąd w kodzie źródłowym i ponownie uruchom program przyciskiem .

#### Zadanie 1.4. Porównanie środowisk programistycznych Dev-C++ i Code::Blocks

- Na podstawie realizacji zadań 1.2 i 1.3 dokonaj porównania procesu edycji i uruchamiania programów w tych środowiskach.
- Zapoznaj się z opiniami programistów na temat wyboru środowiska programistycznego w języku C np. <http://cpp0x.pl/kursy/Kurs-C++/Poziom-1/Wybieramy-srodowisko-pracy/4>.