Wstęp

- Do pisania programów będziemy używali *środowiska* Codeblocks. Jest to bardzo popularne, darmowe środowisko dostępne pod Windowsem i Linuksem
- Powinniśmy stworzyć nowy *projekt*, w nowym folderze. Projekt musi mieć jakąś nazwę. Napisany w danym projekcie program ma taką samą nazwę. Programy, które będziemy pisali to *Console application* - jest to program, który komunikuje się z użytkownikiem przy pomocy klawiatury i tekstowego okienka. Codeblocks tworzy programy w dwóch wariantach: Debug i Release. Na etapie pisania programu będziemy korzystali z wariantu Debug. Wariant Release przeznaczony jest dla ostatecznej wersji programu. Tworzenie wersji Release trwa dłużej, więc jest mniej wygodne, ale tak powstały program jest bardzo zoptymalizowany pod względem szybkości i wykorzystania pamięci.

Wstęp

- Po stworzeniu projektu automatycznie powstaje plik projektu, o nazwie takiej jak projekt i rozszerzeniu `*.cbp`. Ten plik zawiera różne informacje o konfiguracji środowiska Codeblocks. Nie należy go przesyłać do oceny wraz z projektami. Powstaje też główny plik programu, który standardowo nazywa się `main.cpp`. Nazwę tego pliku można zmienić, ale powinna mieć rozszerzenie `*.cpp`
- Program napisany w języku C++ musi mieć dokładnie jedną funkcję o nazwie `main()`. Z reguły zawiera wiele innych funkcji.

Wstęp

- Przykładowy program. Zawartość automatycznie wygenerowanego pliku `main.cpp` należy zastąpić następującą treścią

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Mój pierwszy program" << endl;
    return 0;
}
```

- Wkrótce dokładnie poznamy rolę każdej linijki

Wstęp

- Język C++ stosuje tak zwany wolny format. Układ graficzny kodu *źródłowego*, czyli tego co sami wpisujemy nie ma znaczenia. Poszczególne instrukcje programu oddzielone są średnikami, i tylko to jest ważne dla kompilatora. Natomiast warto trzymać się jakiegoś standardu pisania. Ma to na celu ułatwienie czytania kodu źródłowego nam. Każda instrukcja powinna być w osobnej linijce, kolejne bloki logiczne powinny mieć wcięcia w stosunku do lewego marginesu, powinno się dopisywać komentarze
- Poprawia to czytelność, zwłaszcza w czasie wykonywania programu linijka po linijce (debugging)
- Program może być i najczęściej jest rozłożony na osobne pliki

Wstęp

- Tłumaczenie programu napisanego w C++ na program wykonywalny odbywa się w kilku krokach. Najpierw kod źródłowy przeglądany jest przez tak zwany preprocesor. Szuka on instrukcji zaczynających się znakiem #. To nie są instrukcje języka C++, ale preprocesor jest częścią standardu C++.
- W przykładowym programie powyżej instrukcja `#include <iostream>` informuje preprocesor, że program będzie używał biblioteki `iostream`. W tej bibliotece znajduje się *obiekt* `cout`, którego używamy do wypisania tekstu w okienku.
- Filozofia języka C++ jest taka, że sam język jest bardzo „szczupły", a funkcjonalności zgromadzone są w specjalizowanych bibliotekach, które same też są standardem języka C++. Pisząc programy w C++ stale będziemy więc korzystać z różnych bibliotek.

Wstęp

- Kolejnym krokiem w tworzeniu programu (po przejściu preprocesora) jest *kompilacja*. W tym kroku kod źródłowy tłumaczony jest na kod maszynowy, ale nie jest to jeszcze gotowy program, nie są jeszcze ustalone ostateczne adresy obiektów w pamięci. Poszczególne pliki programu (jeżeli jest więcej niż jeden) kompilowane są osobno. Wynikiem kompilacji jest plik *obiektowy*, który ma taką samą nazwę jak kompilowany plik, z rozszerzeniem `*.o`
- Ostatnim krokiem w tworzeniu programu jest łączenie (linkowanie) skompilowanych plików obiektowych, także bibliotek w ostateczny program.
- Instrukcja `build` wykonuje wszystkie te kroki od razu
- Plik `iostream.h` - nagłówki funkcji bibliotecznych potrzebnych w programie - tylko to potrzebne jest na etapie kompilacji danego pliku
- Same funkcje biblioteczne przechowywane są przez Codeblocks w postaci już skompilowanych plików obiektowych i są dołączane na etapie linkowania

Wstęp

- Wynikiem całego procesu jest plik *wykonywalny*. W przypadku systemu Windows jest to plik o takiej samej nazwie jak projekt, z rozszerzeniem `*.exe`. W przypadku systemu Linux plik nie ma rozszerzenia, ale ma *atrybut* wykonywalności. Żeby taki program uruchomić należy wpisać w wierszu poleceń w konsoli jego nazwę. W przypadku systemu Linux powinno się też dodać ścieżkę `.\*`, gdyż Linux szuka programów do wykonania w różnych specjalnych folderach.
- Kiedy uruchamiamy program ze środowiska Codeblocks okno programu pozostaje po wykonaniu otwarte, żeby umożliwić nam zorientowanie się, co program wypisał. Żeby zamknąć okno programu trzeba wcisnąć dowolny klawisz
- Przykład 1.

Wstęp

- Komentarze są ważne. Powinny stanowić 2/3 kodu źródłowego. Bez dobrych komentarzy bardzo trudno zorientować się w kodzie po jakimś czasie.
- `funt`, `kilogram`, `przelicznik` to zmienne. Przechowują liczby. Mają swoją nazwę, swój typ i swoje miejsce w pamięci. Muszą być zadeklarowane i zdefiniowane. Mogą być „zainicjalizowane". Jeżeli nie są, kompilator nada im początkową wartość 0 (uwaga - nie zawsze).
- Nazwa zmiennej: litery, cyfry, podkreślenia. Nie może się zaczynać od cyfry, nie może się pokrywać z nazwami już zadeklarowanymi w istniejącej przestrzeni nazw. Małe i duże litery są rozróżniane.

Wstęp

- Deklaracja zmiennej: podanie tylko podstawowych informacji - nazwa i typ. Te informacje wystarczają do korzystania ze zmiennej w programie
- Definicja zmiennej: podanie wszystkich informacji, w szczególności gdzie zmienna ma być utworzona i ostatecznie utworzenie zmiennej. Te dodatkowe informacje potrzebne są kompilatorowi do umieszczenia zmiennej w ostatecznym wykonywalnym programie.
- Dla prostych zmiennych z reguły używa się tylko definicji (wyjątek: `extern`). Dla funkcji z reguły najpierw podaje się deklarację, a definicję na samym końcu.