Podstawy programowania w C++

Język naturalny a język programowania

Opracował: Andrzej Nowak

Bibliografia:

CPA: PROGRAMMING ESSENTIALS IN C++ https://www.netacad.com

Wstęp

Każda aktywność twórcza wymaga narzędzi, a programowanie nie jest wyjątkiem.
W najprostszej (nie mówiąc już najbardziej prymitywnej) formie programowanie wymaga kartki papieru i ołówka.

Oczywiście nie będziemy ćwiczyć programowania w ten sposób - ponieważ uruchamianie kodu jest jedyną metodą sprawdzenia, czy jest ona poprawny. Programowanie wymaga komputera wyposażonego w dodatkowe narzędzia.

W tej części poznasz kilka sposobów korzystania z komputera jako stacji roboczej programisty.
Jednak nie zapominaj, że istnieje wiele czynników wpływających na działania, które muszą (lub nie muszą) być wykonywane:

- platforma sprzętowa,
- system operacyjny,
- wersja systemu operacyjnego itp.

Nie znam Twojego systemu, więc ten przewodnik jest ogólny.
Pokaże Ci kierunek - sam musisz znaleźć rozwiązanie.

Niestety wymagany jest dostęp do Internetu.
Musisz pobrać praktycznie wszystkie pliki instalacyjne ze stron producentów oprogramowania.

Przy okazji możesz z powodzeniem uruchamiać wszystkie prezentowane programy i przykłady bez użycia specjalistycznego środowiska - tylko za pomocą standardowego edytora tekstu i narzędzi kompilatora.

Zasadniczo istnieją dwa sposoby uruchamiania kodów programów:

- korzystanie z IDE zainstalowanego lokalnie
- korzystanie z narzędzi on-line

Co to jest IDE?

IDE (***Integrated Development Environment***) to aplikacja, która zazwyczaj składa się z:

- edytora kodu,
- kompilatora,
- debuggera ,
- graficznego interfejsu użytkownika (**GUI -** ang. **graphical user interface**).

Programowanie z IDE ma wiele zalet - dostajesz zestaw narzędzi zawierający wszystko, czego możesz potrzebować.

IDE zapewnia wygodne środowisko programistyczne wyposażone we wszystkie środki, materiały i pomoce.

Są też pewne wady:

Wygodne środowisko programistyczne zajmuje duży obszar pamięci operacyjnej - zużywa wiele zasobów i, szczerze mówiąc, prawdopodobnie nie potrzebujesz większości funkcji, które mogą wykonywać.

Korzystanie z narzędzi on-line umożliwia pisanie, przechowywanie i uruchamianie kodu bez instalowania czegokolwiek.

Wyobraź sobie to jako uproszczone IDE dostępne zdalnie przez Internet.

Oznacza to, że potrzebujesz dwóch rzeczy:

- przeglądarki internetowej,
- dostępu do Internetu.

Jeśli możesz wypróbować oba podejścia, wybierz to, które jest dla Ciebie wygodniejsze.

Jeśli nie możesz - wybierz to, z którego możesz skorzystać.

Wybierz swoje IDE

Na rynku jest wiele IDE, zarówno bezpłatnych, jak i płatnych. Aby dowiedzieć się, jak duża jest lista zintegrowanych środowisk programistycznych dla języka C ++, możesz odwiedzić Wikipedię.

Przedstawiam 5 przykładowych IDE.

Microsoft © Visual Studio Express ®

Jedna platforma (teraz ma nawet wsparcie dla wielu platform) zaprojektowane specjalnie do budowania programów w C ++, zarówno pod, jak i dla systemu operacyjnego MS Windows.

- **strona domowa:** https://www.visualstudio.com
- **downloads:** https://www.visualstudio.com/products/free-developer-offers-vs.aspx
- **licencja:** zastrzeżona, ale ograniczona darmowa wersja o nazwie Visual Studio Community jest dostępna do pobrania; wymaga rejestracji.

Eclipse

Wieloplatformowe środowisko programistyczne zaprojektowane specjalnie dla środowiska Java.

Możliwość programowania w języku C ++ bez dodatkowej konfiguracji (dostępna jest dedykowana wersja C ++ do pobrania).

- **strona domowa:** https://eclipse.org
- **downloads:** https://www.eclipse.org/downloads
- **licencja**: Licencja publiczna Eclipse (bezpłatna i otwarta)

NetBeans

Wieloplatformowe środowisko programistyczne zaprojektowane specjalnie dla środowiska Java.

Możliwość programowania w języku C ++ bez dodatkowej konfiguracji (dostępna jest dedykowana wersja C ++ do pobrania).

- **strona domowa:** https://netbeans.org
- **downloads:** https://netbeans.org/downloads/index.html
- **licencja**: Common Development and Distribution License or GNU Public License version 2 (free and open) Licencja na powszechną dystrybucję i dystrybucję lub GNU Public License wersja 2 (bezpłatna i otwarta)

Code::Blocks

Wieloplatformowe środowisko programistyczne przeznaczone do programowania w języku C / C ++.

Domyślny instalator Windows nie zawiera kompilatora C ++ - zamiast tego użyj nazwy zawierającej "mingw-setup" w nazwie pliku.

- **strona domowa:** http://www.codeblocks.org
- **downloads:** http://www.codeblocks.org/downloads/binaries
- **licencja**: GNU Public License version 3 (free and open) GNU Public License wersja 3 (bezpłatna i otwarta)

XCode

Jednolite środowisko programistyczne zaprojektowane specjalnie do tworzenia aplikacji dla systemów operacyjnych zaprojektowanych przez Apple Inc. Programowanie w C ++ jest w pełni dostępne.

- **strona domowa:** https://developer.apple.com/xcode
- **downloads:** https://developer.apple.com/xcode/download
- **licencja**: zastrzeżone, ale bezpłatne dla użytkowników Mac OS X; zintegrowany z OS X i preinstalowany.

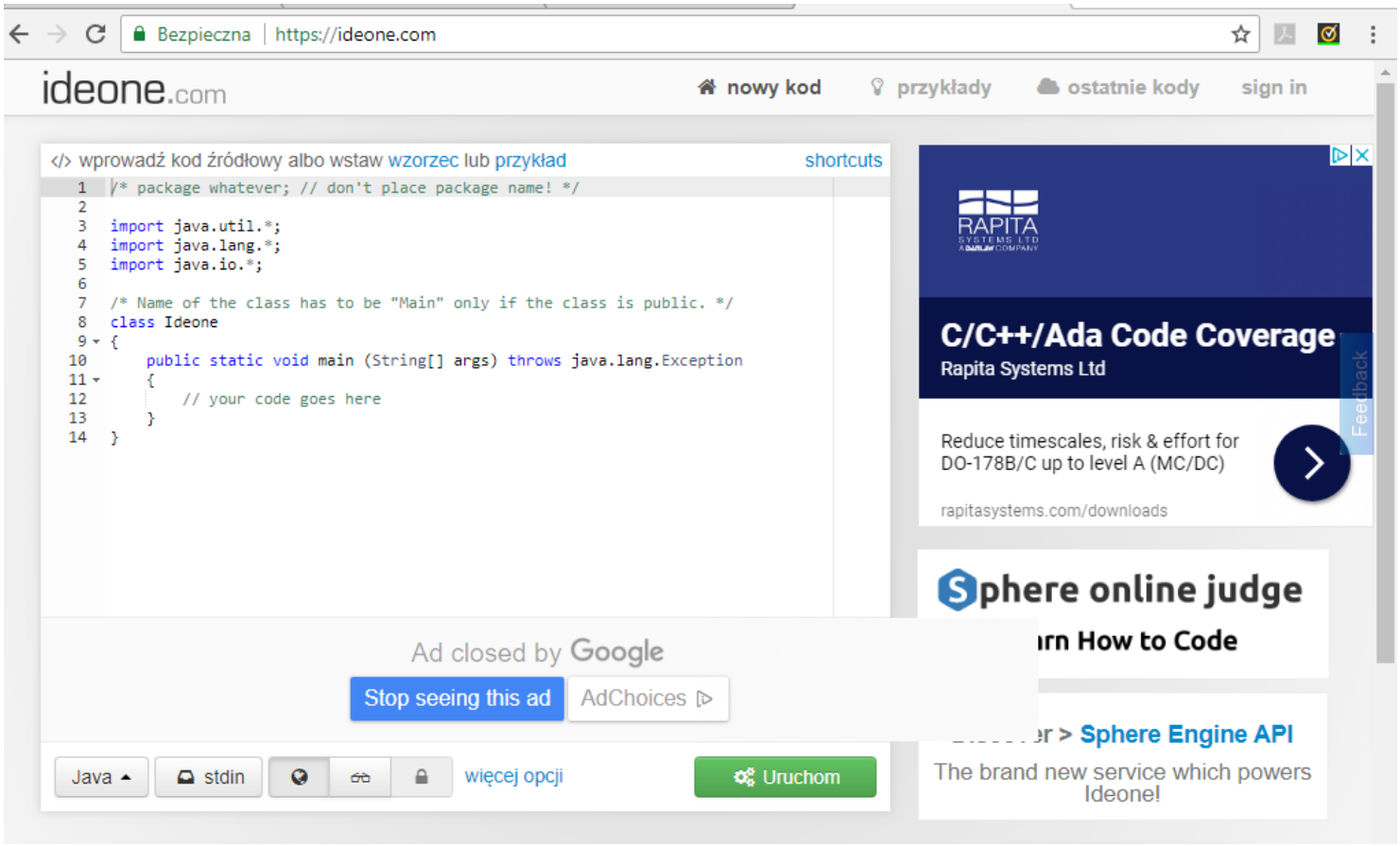
Jeśli jesteś użytkownikiem systemu Linux, spróbuj użyć podstawowych narzędzi systemowych, aby pobrać i zainstalować IDE.

Jeśli używasz innego systemu operacyjnego, wyszukaj kompletny pakiet instalacyjny.

	MS© Windows®	Linux	Mac OS
Visual Studio	✓		
Eclipse	✓	✓	✓
NetBeans	✓	✓	✓
Code::Blocks	✓	✓	✓
XCode			✓

Narzędzia on-line

ideone



Aby zacząć programowanie, nie musisz niczego instalować.

Narzędzie on-line o nazwie **ideone** dostępne w witrynie http://ideone.com.

Chociaż nie musisz się rejestrować, aby rozpocząć pracę, sugerujemy, aby to zrobić - umożliwi to dodatkowe, cenne funkcje.

Rejestracja jest łatwa.

Możesz także użyć swojego konta na Facebooku, aby zalogować się do **ideone** - dzięki temu cały proces będzie jeszcze szybszy i wygodniejszy.

Po zalogowaniu musisz dokonać personalizacji i dwie rzeczy są niezbędne:

- zmień domyślny język programowania na "C ++" (nie zapomnij tego zrobić),
- włącz podświetlanie składni - ułatwi to wykonuj swoją pracę.

Aby przetestować środowisko ideone, przejdź do zakładki "nowy kod" i po prostu skopiuj i wklej następujący tekst w polu kodu źródłowego:

```
#include <iostream>
using namespace std;
int main (void)
{
    cout << "Dziala" << endl;
 }
```

Następnie kliknij przycisk "Ideone it". Tekst "Dziala" powinien pojawić się niemal natychmiast w polu standardowym - oznacza to, że Twój kod źródłowy został szczęśliwie skompilowany i uruchomiony.

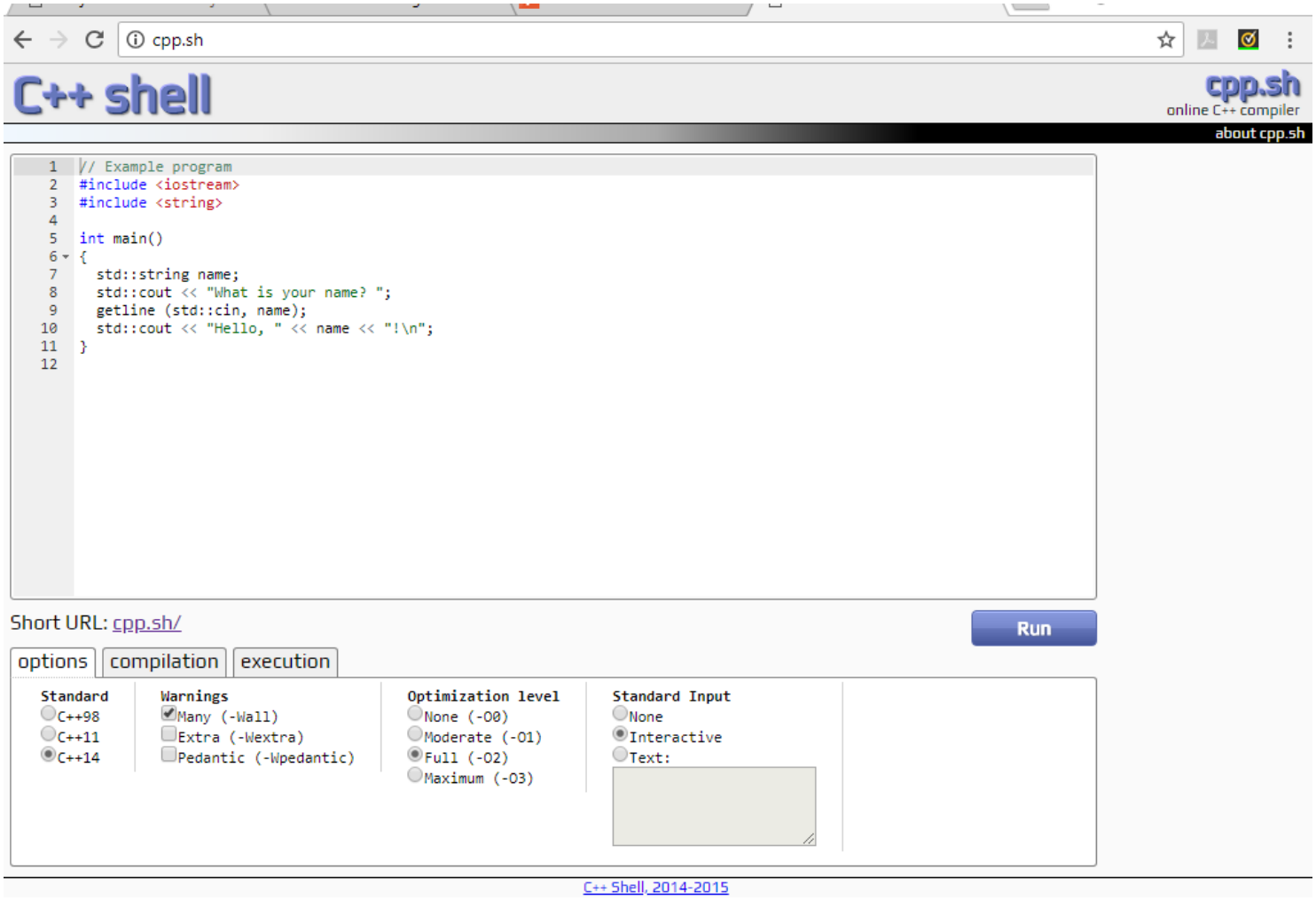
Uwaga:

Nie możesz wysyłać więcej niż 1000 zgłoszeń miesięcznie.

Jeśli twój program odczytuje jakiekolwiek dane od użytkownika, będziesz musiał przygotować dane w polu standardowym przed uruchomieniem kodu (normalnie proces odczytywania danych wejściowych jest interaktywny).

Sprawdź http://ideone.com/faq, aby uzyskać więcej informacji.

C++ shell - cpp.sh



Innym narzędziem on-line o bardzo podobnej funkcjonalności jest **C++ shell** dostępna pod adresem http://cpp.sh.

Nie musisz się rejestrować, a jego główna operacja jest bardzo podobna do **ideone**, chociaż **C++ shell** nie oferuje tylu przydatnych funkcji co **ideone** (np.

- Nie możesz ani zapisać, ani opublikować swojego kodu.
- Nie zawiera również wsparcia (prawdę mówiąc, jest tak prosty w obsłudze, że nie potrzebujesz żadnej pomocy).

Do czego służy język?

Można powiedzieć, że język jest narzędziem do wyrażania i rejestrowania ludzkich myśli.

Innymi słowy, jest to mechanizm znany nam i naszym partnerom, pozwalający nam wszystkim zrozumieć i być zrozumianym. Używamy języka do mówienia, pisania, czytania, słuchania i myślenia.

Przynajmniej jeden język towarzyszy nam przez całe nasze życie - to nasz język ojczysty, którego od samego początku uczymy się niemal nieświadomie. Wielu z nas nauczy się również innych języków, głównie w wyniku świadomej decyzji, wymuszonej warunkami społecznymi lub potrzebami biznesowymi. Języki używane do komunikacji z innymi ludźmi są nazywane językami naturalnymi. Zostały stworzone przez wiele stuleci i wciąż podlegają zmianom. Jeśli zignorujemy języki, które zostały sztucznie utworzone, takie jak esperanto, a nawet quenya (język używany przez elfy w świecie Tolkiena), ich rozwój jest prawie niezależny i ewoluuje w sposób naturalny, w sposób, który nie daje nam żadnej kontroli.

Istnieją jednak języki, których tworzenie i rozwój były (i często pozostają) podyktowane pewnymi specyficznymi potrzebami, a ich rozwój podlega w pełni kontroli bardzo szerokich grup ludzi, takich jak międzynarodowe komitety lub grupy robocze. Kształt tych języków jest zdefiniowany przez międzynarodowe standardy i choć mogą być zrozumiane przez wiele osób, wymiana myśli między ludźmi nie jest ich priorytetem.

Takie języki to między innymi języki programowania. Prawdopodobnie już znasz tę koncepcję. Język programowania definiowany jest przez pewien zestaw sztywnych reguł, znacznie bardziej nieelastyczny niż jakikolwiek język naturalny.

Leksykon - zbiór reguł określających, które symbole (litery, cyfry, znaki interpunkcyjne itd.) mogą być używane w języku.

Syntaktyka (składnia języka) - zestaw zasad określających odpowiednie sposoby zestawiania symboli

Semantyka języka – określa znaczenie każdego stwierdzenia wyrażonego w danym języku.

Każdy program, który piszemy, musi być bezbłędny na trzy sposoby:

- leksykalnie,
- syntaktycznie,
- semantycznie,

w przeciwnym razie program nie będzie działał lub przyniesie niedopuszczalne wyniki.

Możesz być pewien, że napotkasz wszystkie te błędy, ponieważ błądzenie jest ludzkie i to właśnie omylni ludzie piszą programy komputerowe.

Siła ekspresji języków programowania jest znacznie słabsza od języków naturalnych.

Nie możemy (choć możemy próbować) używać takiego języka do wyrażania ludzkich emocji i trudno wyobrazić sobie deklarację miłości zakodowaną w języku programowania.

Dzieje się tak dlatego, że komunikat osadzony w programie komputerowym nie jest przeznaczony dla człowieka, ale dla komputera.

Być może zastanawiasz się, dlaczego w ogóle musimy używać języka programowania i to jest dobre pytanie. Spróbujmy na nie odpowiedzieć.

Co to jest język wysokiego poziomu?

Komunikowanie się z komputerem.

Komputer, nawet najbardziej zaawansowany technicznie, pozbawiony jest nawet śladu inteligencji.

Można powiedzieć, że jest to dobrze wyszkolony pies - odpowiada tylko na określony zestaw znanych poleceń (ang. **set of known commands**). Czasami, jak pies, po prostu się zamyka i odmawia wykonania tego, co mu powiedziano.

Rozpoznane polecenia są bardzo proste. Możemy sobie wyobrazić, że komputer odpowiada na polecenia typu "weź ten numer, dodaj do drugiego i zapisz wynik".

Kompletny zestaw dobrze znanych poleceń nazywany jest **listą instrukcji**, czasami skracaną do **IL**. Różne typy komputerów mogą się różnić w zależności od wielkości ich **IL**, a same instrukcje mogą być całkowicie różne w różnych modelach.

IL (ang. **instruction list**) **lista instrukcji -** jest w rzeczywistości alfabetem powszechnie znanym jako język maszynowy (ang. **machine language**). Jest to najprostszy i najbardziej podstawowy język, z którego możemy korzystać, aby wydawać polecenia dla naszego komputera. Można powiedzieć, że jest to język ojczysty komputera.

Programowanie komputerowe jest aktem komponowania wybranych poleceń (instrukcji) w kolejności, która wywoła pożądany efekt. Sam efekt może być różny w każdym przypadku - zależy od intencji programisty, wyobraźni, wiedzy i doświadczenia.

> Jest możliwe (i to często dzieje się w praktyce), aby program komputerowy był kodowany bezpośrednio w języku maszynowym za pomocą podstawowych instrukcji (zamówień).

Ten rodzaj programowania jest żmudny, czasochłonny i wysoce podatny na błędy programisty. Na wczesnych etapach technologii komputerowej była to jedyna dostępna metoda programowania i bardzo szybko ujawniła ona swoje poważne wady. Programowanie w języku maszynowym wymaga pełnej znajomości projektu sprzętu komputerowego i jego wewnętrznej struktury. Oznacza to również, że zastąpienie komputera innym, który różni się od swojego poprzednika, może spowodować, że cała wiedza programisty stanie się bezużyteczna. Również stare programy mogą być zupełnie bezużyteczne, jeśli nowy komputer użyje innej IL. Zatem program napisany dla określonego typu komputera może być całkowicie bezużyteczny dla innych komputerów i na odwrót. Po drugie, programy napisane w języku maszynowym są bardzo trudne do zrozumienia dla ludzi, w tym doświadczonych programistów. Jest również tak, że opracowanie programu w języku maszynowym zajmuje dużo czasu i jest bardzo kosztowne i kłopotliwe.

Wszystkie te okoliczności prowadzą do potrzeby pewnego rodzaju pomostu między językiem ludzi (język naturalny) a językiem komputerowym (język maszynowy).

Ten most jest również językiem - pośrednim wspólnym językiem dla ludzi i współpracujących ze sobą komputerów.

Taki język jest często nazywany językiem programowania wysokiego poziomu.

Język taki jak ten jest co najmniej w pewnym stopniu podobny do języka naturalnego - używa:

- symboli,
- słów
- konwencji czytelnych dla ludzi.

Taki język umożliwia ludziom wyrażanie złożonych poleceń dla komputerów, dlatego często nazywany jest – językiem wysokiego poziomu.

Programy napisane w językach wysokiego poziomu mogą być tłumaczone na dowolną liczbę różnych języków maszynowych, dzięki czemu można je wykorzystywać na wielu różnych komputerach. Nazywa się to przenośnością.

Co to jest i do czego służy kompilator?

Kompilator – wyspecjalizowany program komputerowy służący do tłumaczenia z języka wysokiego poziomu na język maszynowy.

Wróćmy teraz do bardziej interesujących zagadnień związanych z procesem tworzenia nowego programu.

Wiemy już, że naszym głównym zadaniem jest napisanie programu zgodnie z zasadami wybranego języka programowania.

Ten program (który w rzeczywistości jest tylko tekstem) nazywany jest **kodem źródłowym** lub po prostu źródłem, podczas gdy plik zawierający źródło nazywany jest plikiem źródłowym.

Do napisania kodu źródłowego potrzebny jest edytor tekstu, który umożliwia manipulowanie tekstem bez informacji o formatowaniu (z tego powodu Microsoft Word nie jest dobrym wyborem - Notatnik jest lepszy).

Ten kod znajduje się w pliku, a nazwa pliku powinna sugerować jego zawartość.

Na przykład często plik zawierający kod źródłowy w języku C ++ ma nazwę kończącą się **sufiksem** **`.cpp`**, więc jeśli napiszesz program komputerowy i zdecydujesz się nazwać go `programik`, dobrym pomysłem byłoby umieszczenie kod źródłowy do pliku o nazwie `programik.cpp`.

Uwaga:

> Niektóre platformy mogą preferować inne sufiksy, takie jak **`cc`**, **`cp`**, **`cxx`**, **`c ++`** lub nawet **`C`** (zauważ, że jest to wielka litera). Aby uzyskać szczegółowe informacje, zapoznaj się z dokumentacją kompilatora.

Następnie musisz skompilować swój kod źródłowy.

> Aby to zrobić, musisz uruchomić odpowiedni kompilator i poinstruować go, gdzie zapisałeś kod źródłowy, który chcesz przetłumaczyć na język maszynowy.

Kompilator odczyta kod, wykona skomplikowaną analizę, a następnie określi, czy zostały popełnione błędy podczas procesu kodowania..

Jeśli kompilator nie znajdzie żadnych błędów w Twoim źródle, wynikiem będzie plik zawierający Twój program przetłumaczony na język maszynowy.

Ten plik jest zwykle nazywany plikiem wykonywalnym.

Nazwa pliku zależy od używanego kompilatora i systemu operacyjnego, z którym współpracujesz.

Na przykład większość kompilatorów zaprojektowanych dla systemu Unix / Linux domyślnie tworzy plik wyjściowy o nazwie "a.out".

Kompilatory zaprojektowane do użycia w systemie MS Windows® mogą nadać temu plikowi tę samą nazwę, co plik źródłowy, a jedynie zmieniać przyrostek z pliku .cppto .exe.

Oczywiście cały proces jest nieco bardziej skomplikowany.

Twój kod źródłowy może być ogromny i podzielony na kilka lub nawet dziesiątki plików źródłowych.

Możliwe też, że program nie został napisany przez ciebie samego, ale przez cały zespół, w którym to przypadku podział źródeł na wiele plików jest po prostu koniecznością.

W tym przypadku proces kompilacji dzieli się na dwie fazy:

- kompilację źródła w celu przetłumaczenia go na język maszynowy,
- połączenie (lub sklejenie) kodu wykonywalnego z kodem wykonawczym uzyskanym od innych programistów w jeden i ujednolicony produkt.

Faza "sklejania" różnych kodów wykonywalnych jest powszechnie znana jako łączenie, podczas gdy program, który przeprowadza proces, nazywany jest łącznikiem.