Wstęp do programowania

- Wprowadzenie
- Film samouczek
- Przeczytaj
- Sprawdź się
- Dla nauczyciela

Bibliografia:

- Źródło: *Czym jest programowanie?*, dostępny w internecie: kft.umcs.lublin.pl [dostęp 17.05.2021].
- Źródło: Daniel McCracken, William I. Salmon, *A Second Course in Computer Science with Modula-2 [data structures]*, 1987, dostępny w internecie: kft.umcs.lublin.pl [dostęp 17.05.2021].

Źródło: Danielle MacInnes, domena publiczna.

Współcześnie trudno sobie wyobrazić, jak moglibyśmy funkcjonować bez programowania. Wpływa ono na niemal każdą dziedzinę życia. Poznaliśmy już najpopularniejsze języki programowania oraz sposoby ich klasyfikacji. W tym e-materiale poznamy etapy powstawania programu komputerowego, a następnie napiszemy przykładowy program.

Jak wyglądają środowiska programistyczne poszczególnych języków programowania dowiesz się w pozostałych e-materiałach z tej serii:

- Wstęp do programowania w języku C++,
- Wstęp do programowania w języku Java,
- Wstęp do programowania w języku Python.

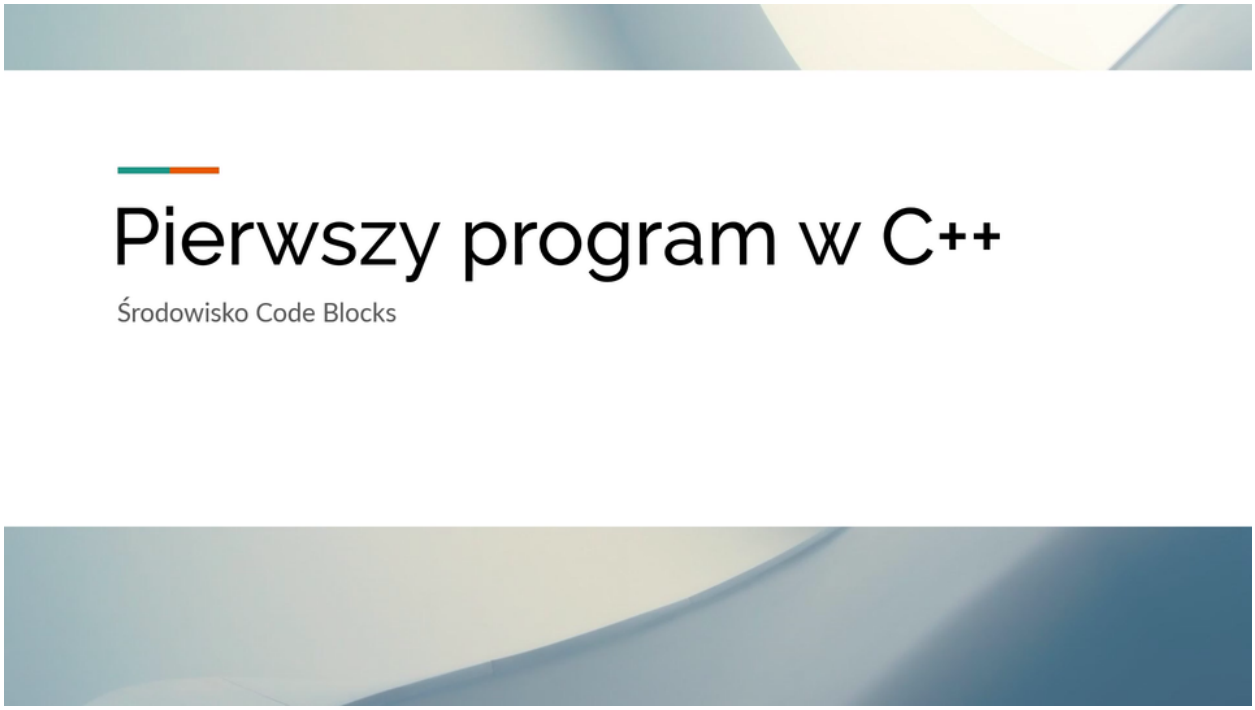
> **Twoje cele**

- Prześledzisz, w jaki sposób zainstalować i skonfigurować środowisko programistyczne Code::Blocks.
- Przeanalizujesz budowę szkieletu programu napisanego w języku C++.
- Wyjaśnisz, czym jest kompilator, kod źródłowy, konsola.

Film samouczek

Polecenie 1

Zapoznaj się z filmem, który objaśnia, jak skonfigurować środowisko programistyczne Code::Blocks oraz pokazuje, jak napisać pierwszy program w języku C++.



Film dostępny pod adresem /preview/resource/R1I0saV2w18QS

Film przedstawia wprowadzenie do programowania w języku C++.

Polecenie 2

Przygotuj notatkę podsumowującą najważniejsze informacje przedstawione w filmie.

Polecenie 3

Przygotuj notatkę na temat tego, jak wyglądałyby analogiczne programy zapisane w językach Java oraz Python.

Przeczytaj

Program komputerowy

Programem komputerowym nazywamy ciąg instrukcji realizujący dany algorytm – instrukcje te są zapisane za pomocą określonego języka (lub języków) programowania.

Program to wynik końcowy szeregu czynności, które należy wykonać, aby uzyskać finalną formę.

> **Ważne!**
>
> Proces tworzenia programu można przedstawić w kilku krokach.
>
> 1. Opracowanie kodu źródłowego.
> 2. Zapisanie kodu źródłowego w pliku (lub wielu plikach) o rozszerzeniu zgodnym z danym językiem programowania.
> 3. Kompilowanie pliku z kodem źródłowym.

Czy wiesz, czym są **kod źródłowy** oraz **kompilacja**?

Omówmy każdy z etapów powyżej opisanego procesu.

Etap I: Opracowanie kodu źródłowego

Kod źródłowy to zapis programu komputerowego za pomocą danego języka programowania. Jest to postać programu zrozumiała dla programisty. Kod źródłowy zawiera zdefiniowane operacje, które ma wykonywać program. Są one zapisanie zgodnie z instrukcjami, słowami kluczowymi, zasadami składniowymi oraz zgodnie z leksyką określoną dla wybranego języka programowania.

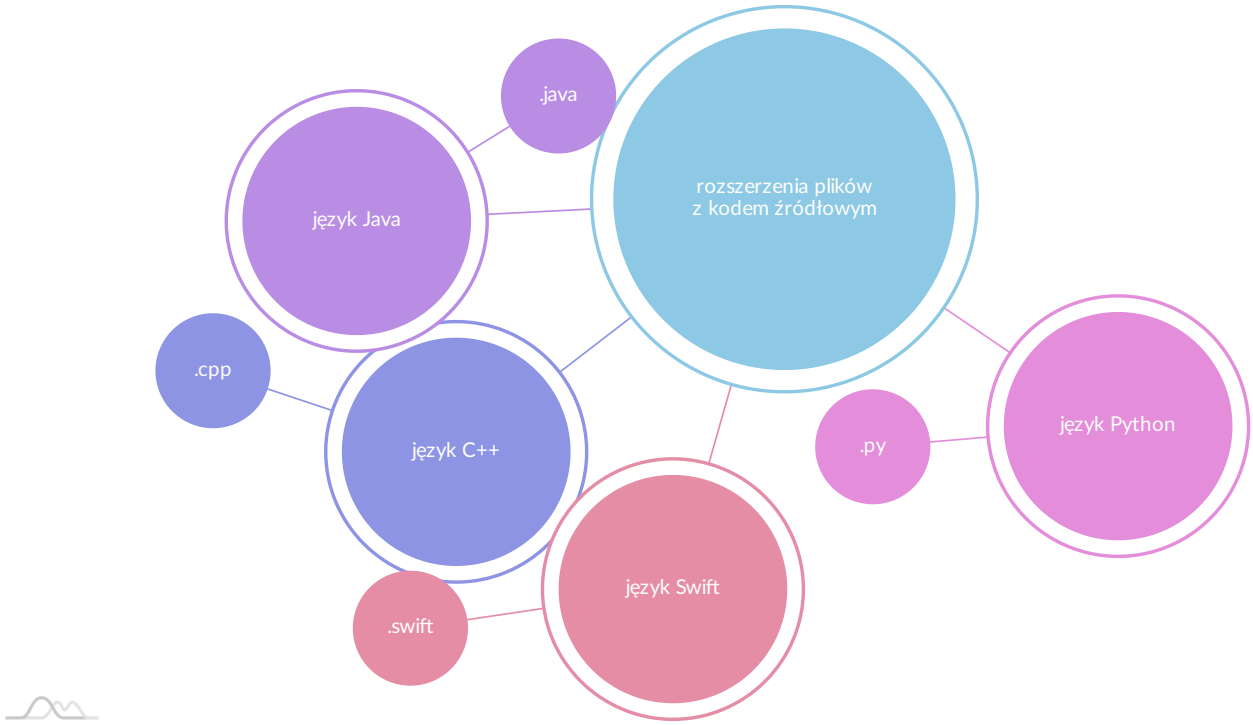
Oto przykład fragmentu kodu źródłowego języka programowania C++:

```
int z = 0;
for (int i = 97; i < 123; i++) {
    alfabet[z] = (char)i;
    z++;
}

for (int z = 0; z < 26; z++) {
    for (int x = 0; x < 26; x++) {
        tabela[z][x] = alfabet[x];
    }

    for (int b = 0; b < 26 - 1; ++b) {
        std::swap(alfabet[b], alfabet[b+1]);
    }

}

for (int t = 0; t < 26; t++) {
    for (int y = 0; y < 26; y++) {
        std::cout << tabela[t][y];
    }
    std::cout << std::endl;
}
```

Opracowanie kodu źródłowego to pierwszy, a zarazem najważniejszy etap tworzenia programu. To w nim zawiera się cała logika programu, realizująca dany algorytm.

Etap II: Plik z kodem źródłowym

Kolejnym etapem jest zapisanie utworzonego wcześniej kodu źródłowego w pliku o odpowiednim rozszerzeniu. Dla każdego języka programowania rozszerzenie jest inne. Oto rozszerzenia plików z kodem źródłowym dla różnych języków programowania:



Etap III: Kompilacja pliku z kodem źródłowym

Kod źródłowy jest niezrozumiały dla komputera, który posługuje się językiem złożonym z zer i jedynek, czyli z liczb binarnych. Dlatego też algorytm musi zostać przekształcony (poddany **translacji**) w **kod wynikowy**, czyli takie instrukcje, które procesor jest w stanie zrozumieć i wykonać. Zadanie to realizuje translator. Istnieją dwie odmiany translatorów: interpretery i kompilatory.

Program zwany **kompilatorem** automatycznie przeprowadza opisaną translację. Współczesne narzędzia do kompilacji zajmują się również asemblacją. Wynikiem działania kompilatora jest utworzenie pliku wykonywalnego, który można uruchomić na innym komputerze bez użycia środowiska programistycznego.

> **Ważne!**
>
> Innym rodzajem translatora jest **interpreter** tłumaczący kod źródłowy na kod maszynowy. Wykonuje on instrukcję po instrukcji – odczytuje kolejne linie kodu, zamienia je na ciągi jedynek i zer, a następnie wysyła do procesora. Gdy ten

> wykona polecenie, interpreter może przetłumaczyć kolejną instrukcję. Interpreter pozwala na kontrolowanie działania programu, ale nie tworzy pliku wykonywalnego, który da się uruchomić na innym komputerze beż użycia interpretera.

Zintegrowane środowisko programistyczne

Wszystkie wyżej opisane czynności najczęściej wykonuje się w narzędziu zwanym zintegrowanym środowiskiem programistycznym (ang. **IDE** – *integrated development environment*). Jest to program lub zespół programów, w którym możemy tworzyć pliki źródłowe, kompilować je oraz uruchamiać. Ponadto niektóre IDE umożliwiają testowanie kodu oraz debugowanie, czyli wyszukiwanie błędów w kodzie.

IDE to podstawowe narzędzie pracy każdego programisty. Dzięki wbudowanym funkcjom, takim jak:

- podpowiadanie składni,
- automatyczne sprawdzanie poprawności kodu,
- posiadanie wbudowanej konsoli

IDE pozwala na skuteczne i efektywne pisanie kodu.

Przejdź do kolejnej sekcji e-materiału, w której dowiesz się, jak skonfigurować Code::Blocks oraz napiszesz swój pierwszy program: Hello World.

Słownik

> **Code::Blocks**
>
> zintegrowane środowisko programistyczne dla języka C++, rozpowszechniane na licencji *Open Source*

> **Hello World**
>
> program, którego jedynym celem jest wypisanie na standardowym wyjściu napisu „Hello World!” lub innego prostego komunikatu; program taki ma na celu

demonstrację języka, środowiska bądź biblioteki, w której był napisany

liczby binarne

liczby, które są zapisywane przy użyciu dwóch cyfr: 0 oraz 1; podstawą systemu binarnego jest liczba 2; systemem takim posługują się m.in. procesory

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zaznacz poprawną odpowiedź. Wskaż, co oznacza akronim IDE.

- ◯ Integrated Develop Environment
- ◯ Integrated Development Environment
- ◯ International Development Environment
- ◯ Installed Development Environment

Ćwiczenie 2

Uzupełnij definicję kodu źródłowego.

Kod źródłowy to zapis [ ] komputerowego za pomocą [ ] [ ], w postaci [ ] dla [ ].

procesora | ciągu poleceń | programisty | programowania | programu | zrozumiałej | operacji | języka

Ćwiczenie 3

Zaznacz wszystkie poprawne odpowiedzi. Wskaż, jakie funkcje może posiadać środowisko programistyczne.

- [ ] automatyczne wykrywanie błędów w kodzie
- [ ] podpowiadanie kodu
- [ ] wbudowana konsola
- [ ] debugowanie kodu

Ćwiczenie 4

Uporządkuj – w poprawnej kolejności – etapy procesu powstawania programu komputerowego.

- Kompilacja pliku z kodem źródłowym.
- Opracowywanie kodu źródłowego.
- Utworzenie pliku z kodem źródłowym.

Ćwiczenie 5

Wskaż, czym jest `#include`.

- ( ) zmienną
- ( ) funkcją
- ( ) dyrektywą
- ( ) biblioteką

Ćwiczenie 6

Zaznacz, które zdania są prawdziwe, a które fałszywe.

Stwierdzenie	Prawda	Fałsz
Funkcja `main` jest główną funkcją programu.	◯	◯
Funkcja `main` jest funkcją, od której zaczyna się wykonywanie kodu.	◯	◯
Funkcja `main` jest funkcją, która wykonuje się jako ostatnia przed zakończeniem programu.	◯	◯
Funkcja `main` jest najważniejszą funkcją w programie.	◯	◯

Ćwiczenie 7

Dokończ zdanie.

Typ integer (`int`) służy do przechowywania...

- ◯ liczb zmiennoprzecinkowych.
- ◯ liczb rzeczywistych.
- ◯ ciągów znaków.
- ◯ liczb całkowitych.

Ćwiczenie 8

Zapoznaj się z tekstem i wykonaj ćwiczenie.

Czym jest programowanie?

Nazwy zmiennych w programie. Dobrze jest stosować **rzeczowniki** dla oznaczania nazw wszelkich danych, **czasowniki** dla funkcji i procedur (rozkazy, np. dziel, dodaj itp), **przymiotniki** dla zmiennych logicznych (w Pascalu **boolean**). Jednolite nazwy są mało czytelne. Najlepiej jest używać nazw, które coś oznaczają, powiadamiając tym samym użytkownika (intencjonalnie) o możliwościach funkcji czy procedury, czy też wskazując sens zmiennych, itd. Nazwy bardzo długie, chociaż mogą być komunikatywne, są mało praktyczne jeśli nie stosujemy wyróżnień takich jak duże litery czy podkreślenie (*underscore*) np. nazwa `dodajdwieliczby` jest mniej czytelna od nazwy `DodajDwieLiczby` lub od nazwy `dodaj_dwie_liczby`.

Źródło: *Czym jest programowanie?*, dostępny w internecie: kft.umcs.lublin.pl [dostęp 17.05.2021].

Daniel McCracken, William I. Salmon

A Second Course in Computer Science with Modula-2 [data structures]

Nie istnieją żadne absolutne reguły stylu programowania tak samo jak stylu pisania. ponieważ programowanie jest częściowo sztuką, a częściowo nauką.

Źródło: Daniel McCracken, William I. Salmon, *A Second Course in Computer Science with Modula-2 [data structures]*, 1987, dostępny w internecie: kft.umcs.lublin.pl [dostęp 17.05.2021].

Wyjaśnij, jak postrzegane są reguły programowania.

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.

Przedmiot: Informatyka

Temat: Wstęp do programowania

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Prześledzisz, w jaki sposób zainstalować i skonfigurować środowisko programistyczne Code::Blocks.
- Przeanalizujesz budowę szkieletu programu napisanego w języku C++.
- Wyjaśnisz, czym jest kompilator, kod źródłowy, konsola.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- telefony z dostępem do internetu.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Wstęp do programowania". Uczniowie zapoznają się z treściami w sekcji „Film samouczek".

Faza wstępna:

1. Nauczyciel wyświetla temat oraz cele zajęć, omawiając lub ustalając razem z uczniami kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytanie dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu, opartego o programowanie.

Faza realizacyjna:

1. **Praca z tekstem.** Jeżeli przygotowanie uczniów do lekcji jest niewystarczające, nauczyciel prosi o indywidualne zapoznanie się z treścią zawartą w sekcji „Film samouczek". Każdy uczestnik zajęć podczas cichego czytania wynotowuje najważniejsze kwestie poruszane w tekście.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Film samouczek". Uczniowie wspólnie zapoznają się z treścią zawartego w niej multimedium, w którym przedstawione zostało, jak skonfigurować środowisko programistyczne Code::Blocks oraz jak napisać pierwszy program w języku C++.

3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1-8. Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).

Praca domowa:

1. Uczniowie opracowują FAQ (minimum 3 pytania i odpowiedzi) do tematu lekcji („Wstęp do programowania").

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Treści w sekcji „Film samouczek" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.