DO NOWEJ PODSTAWY PROGRAMOWEJ

Część 1

PODRĘCZNIK dla szkół ponadpodstawowych

Informatyka Europejczyka

Zakres podstawowy

Danuta Korman
Grażyna Szabłowicz-Zawadzka

Helion EDUKACJA

Redaktor prowadzący: Joanna Zaręba

Ilustracja na okładce została wykorzystana za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie?ieppp1
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-5704-4

Printed in Poland.

Spis treści

Wstęp

Technologia informacyjna i informatyka to dziedziny, których wykorzystanie i dostępność stale wzrastają, a tempo zachodzących w nich zmian jest nieporównywalne z tempem rozwoju innych dyscyplin. Widać to również w dziedzinie edukacji informatycznej. **Obszary informatyki, które pojawiają się w edukacji informatycznej** na poziomie szkoły podstawowej i ponadpodstawowej, to:

- rozumienie, analizowanie i rozwiązywanie problemów;
- programowanie, aplikacje i robotyka;
- komputery, sieci i urządzenia cyfrowe;
- kompetencje społeczne;
- prawo i bezpieczeństwo.

W konsekwencji na wszystkich etapach realizowane są zagadnienia związane z algorytmiką i programowaniem, co wpływa na rozwój logicznego i algorytmicznego myślenia, a tym samym na umiejętność samodzielnego rozwiązywania problemów.

Informatyka realizowana w szkole ponadpodstawowej na poziomie podstawowym:

- jest przedmiotem obowiązkowym prowadzonym we wszystkich klasach;
- nie jest przedmiotem maturalnym;
- stanowi poziom podstawowy dla przedmiotu informatyka na poziomie rozszerzonym, a więc wchodzi w zakres wymagań maturalnych tego przedmiotu.

Informatyka Europejczyka **to podręczniki do informatyki na poziomie podstawowym** (trzy części) i **na poziomie rozszerzonym** (dwie części). Podręczniki te są w pełni zgodne **z podstawą programową obowiązującą od 2019 roku**, z uwzględnieniem jej specyficznych cech, takich jak:

- **Spiralność** — na każdym etapie wymaga się umiejętności zdobytych wcześniej i rozszerza się je o umiejętności nowe; na początku każdego rozdziału w podręczniku podane są informacje na temat tego, co uczeń powinien pamiętać ze szkoły podstawowej, czego nauczy się z tego podręcznika oraz co zaplanowane jest jako kontynuacja na poziomie rozszerzonym.
- **Myślenie komputacyjne** — w podręczniku wykorzystywane są narzędzia informatyczne do rozwiązywania problemów wywodzących się z różnych dziedzin życia i przedstawiane są różnorodne zastosowania metod i technik algorytmicznych; zastosowano nauczanie poprzez świadome wykorzystanie metod i narzędzi informatycznych.
- **Rozwiązywanie problemów** — zastosowano nauczanie poprzez rozwiązywanie problemów z różnych dziedzin życia.
- **Stopniowe wprowadzanie trudnej problematyki** — abstrakcyjne myślenie algorytmiczne i programowanie kształtowane są stopniowo, od pierwszej klasy szkoły podstawowej, poprzez poziom podstawowy, a dla niektórych również rozszerzony, w szkole ponadpodstawowej, aż do egzaminu maturalnego z informatyki.
- **Praca zespołowa i metoda projektów** — preferuje się pracę w grupach, wspólne rozwiązywanie problemów i uzyskiwanie pozytywnych efektów.

- **Nowoczesność** — uwzględniane są najnowsze trendy w zastosowaniach informatyki i wyborze narzędzi.

Podręcznik powinien być realizowany **w układzie liniowym** — temat po temacie. Jednak jeden temat można wprowadzać na więcej niż jednej lekcji, co zależy od tempa pracy uczniów i konieczności wykonania dodatkowych ćwiczeń utrwalających. Podręcznik zawiera **wiele przykładów i ćwiczeń**, na podstawie których uczeń może przystąpić do **rozwiązywania zadań**. Każdy rozdział kończy się **zadaniami podsumowującymi** przerobiony materiał.

W części pierwszej podręcznika uczeń:

- Poznaje podstawy **algorytmiki i języka tekstowego Python**.
- Uczy się postępowania zgodnie z regulacjami prawnymi dotyczącymi **ochrony danych osobowych** oraz **prawa autorskiego**, poznaje konsekwencje łamania tych zasad.
- Projektuje modele dwuwymiarowe i **trójwymiarowe**, tworzy i edytuje projekty w **grafice rastrowej** i **grafice wektorowej**, wykorzystuje różne formaty obrazów, przekształca pliki graficzne, uwzględniając wielkość i jakość obrazów.
- Opracowuje dokumenty o różnorodnej tematyce, w tym informatycznej, i o rozbudowanej strukturze, posługując się przy tym **konspektem dokumentu**, dzieli tekst na **sekcje** i **kolumny**, tworzy **spisy treści**, **rysunków** i **tabel**, stosuje **własne style** i **szablony**, pracuje nad **dokumentem w trybie recenzji**.

Danuta Korman
Grażyna Szabłowicz-Zawadzka

Rozdział 1.

ROZUMIENIE I ANALIZOWANIE PROBLEMÓW. WPROWADZENIE DO PROGRAMOWANIA W JĘZYKU PYTHON

Co już potrafisz?

Wiesz, czym jest **algorytm**. Potrafisz podawać przykłady algorytmów i analizować ich przebieg.

Określasz **specyfikację analizowanego problemu**, czyli dane i wyniki dotyczące rozwiązywanego zadania.

Potrafisz przedstawiać algorytmy w postaci **schematów blokowych** i **list kroków**. Określasz kroki realizacji algorytmów.

Konstruujesz proste **algorytmy liniowe** i **iteracyjne**.

Wiesz, czym jest **tekstowy język programowania**, i znasz podstawy takiego języka, na przykład Pythona, C++, Processingu, Javy czy Pascala.

Projektujesz, tworzysz i testujesz **oprogramowanie sterujące robotem** na ekranie lub w rzeczywistości.

Czego się nauczysz?

Poznasz wszystkie **etapy rozwiązywania problemów za pomocą komputera** — jednym z nich jest specyfikacja problemu, którą już określasz.

Nauczysz się **klasyfikować tekstowe języki programowania**.

Poznasz **środowisko programowania dla języka Python**.

Dowiesz się, czym jest **rekurencja**, i będziesz ją stosować w algorytmach.

Będziesz **sprawdzać poprawność algorytmów poprzez testowanie** dla przykładowych danych.

Rozpoczniesz **naukę języka Python** — poznasz strukturę programu, operacje wejścia-wyjścia, zmienne, wyrażenia i operatory, instrukcje warunkowe i iteracyjne oraz funkcje.

Co pojawi się na poziomie rozszerzonym?

Poznasz **tekstowy język programowania dopuszczony do egzaminu maturalnego** z informatyki rozszerzonej. Może to być również język Python, którego podstawy poznajesz na lekcjach informatyki na poziomie podstawowym.

Będziesz omawiać oraz stosować **zaawansowane algorytmy iteracyjne** i **rekurencyjne**.

Poznasz i będziesz wykorzystywać reprezentację algorytmów w postaci **pseudokodu**.

Zajmiesz się **analizą efektywności algorytmów**.

Poznasz **źródła błędów pojawiających się w obliczeniach komputerowych**.

Temat 1. Etapy rozwiązywania zadań za pomocą komputera

Wiesz już, że:

- **informatyka** to dziedzina, której głównym celem jest rozwiązywanie problemów z wykorzystaniem komputera;
- **algorytmika** to dział informatyki obejmujący algorytmy oraz ich własności;
- **algorytm** jest dokładnym przepisem na rozwiązanie problemu lub osiągnięcie jakiegoś celu, realizowanym w skończonej liczbie kroków; istnieje wiele sposobów rozwiązania danego zadania, dlatego każdemu problemowi odpowiada wiele metod prowadzących do prawidłowych wyników.

Aby znaleźć rozwiązanie określonego problemu, musisz posiadać informacje początkowe, które następnie są wykorzystywane przy realizacji algorytmu prowadzącego do otrzymania danych wyjściowych. Dlatego mówimy również, że algorytm to skończona liczba wykonywanych kolejno czynności, które prowadzą do przekształcenia danych wejściowych na wyniki będące rozwiązaniem danego zadania.

Ciekawostka

Termin **algorytm** pochodzi od nazwiska arabskiego matematyka i astronoma Muhammada ibn Musa Al-Chorezmiego, żyjącego na przełomie VIII i IX wieku. Zapoczątkował on stosowanie metod obliczeniowych w matematyce, ponadto upowszechnił wykorzystanie systemu dziesiętnego i określił znaczenie zera w obliczeniach.

Przykład 1.1.

Przygotowywanie potraw odbywa się według dokładnej instrukcji. Wiemy, jakie składniki są nam potrzebne oraz co chcemy przygotować — to specyfikacja naszego zadania. Wiemy również, jakie czynności krok po kroku trzeba wykonać, aby uzyskać oczekiwany efekt — to lista kroków lub opis słowny algorytmu. Z algorytmami spotykasz się każdego

dnia, definiujesz ich specyfikację, podając dane i wyniki, ponadto zapisujesz je w sposób czytelny i jednoznaczny, najczęściej jako listę kroków. Każda instrukcja to zapis algorytmu.

Ćwiczenie 1.1.

Podaj przykłady algorytmów związanych z życiem codziennym lub innymi — poza informatyką — przedmiotami szkolnymi. Określ specyfikację tych algorytmów, czyli opisz dane i wyniki. Podaj krok po kroku czynności, które trzeba wykonać, aby otrzymać oczekiwany wynik.

Etapy rozwiązywania zadań

Realizacja zadań za pomocą komputera przebiega etapami. Pierwszy etap — specyfikacja zadania — jest Ci już znany ze szkoły podstawowej. Poniżej wyszczególnione są wszystkie etapy, które prowadzą do rozwiązania problemu z wykorzystaniem komputera.

1. **Określenie problemu, czyli specyfikacji zadania**:
   - **dane** (dane wejściowe),
   - **wyniki** (dane wyjściowe).

   Określając dane wejściowe i wyniki, podajemy również ich typ (czyli określamy rodzaj danych, na przykład liczby całkowite, liczby rzeczywiste, znaki, teksty) oraz sposób prezentacji. Ważny jest ponadto związek między nimi, a więc wyszczególnienie warunków, jakie muszą spełniać.

2. **Definicja modeli i pojęć oraz zbadanie, czy analizowany problem ma rozwiązanie. Na podstawie uzyskanych informacji wybieramy odpowiedni algorytm rozwiązujący problem**. Istnieje wiele sposobów wykonania danego zadania, z czego wynika, że można skonstruować wiele algorytmów rozwiązujących określony problem. Na tym etapie analizujemy problem i dokonujemy wyboru algorytmu, który ma prowadzić do rozwiązania zgodnego ze specyfikacją.

3. **Zapisanie algorytmu w wybranej postaci**:
   - opisu słownego,
   - listy kroków,
   - schematu blokowego,
   - pseudokodu,
   - drzewa algorytmu,
   - drzewa wyrażenia,
   - programu
   - i innych.

   Jeśli chcemy użyć komputera, musimy skonstruować algorytm w postaci programu w wybranym języku programowania. Pozostałe formy mogą, ale nie muszą powstać jako pomoc w procesie pisania programu.

   W szkole podstawowej pojawiły się już: opis słowny algorytmu, lista kroków, schemat blokowy oraz zapis algorytmu w postaci programów w wybranych przez nauczyciela wizualnych i tekstowych językach programowania.

4. **Testowanie rozwiązania poprzez wykonywanie obliczeń na komputerze**, w tym testowanie programu, który jest implementacją algorytmu, dla różnych danych.

Algorytm powinien działać dla dowolnych, zgodnych ze specyfikacją, danych wejściowych, dając poprawne wyniki. Jest to jeden ze sposobów sprawdzania poprawności wybranego algorytmu. Algorytm jest poprawny, jeśli dla każdych danych wejściowych jest skończony, a uzyskane wyniki są poprawne, czyli zgodne ze specyfikacją zadania.

5. **Analiza własności wybranego algorytmu**, w tym ocena efektywności przyjętego rozwiązania, złożoności obliczeniowej (czasowej — określającej czas działania algorytmu — i pamięciowej — określającej, ile pamięci potrzeba do realizacji danego algorytmu) oraz błędów zaokrągleń wynikających z obliczeń na liczbach przybliżonych. Wybierany jest algorytm, który nie tylko daje poprawne dane wyjściowe, ale również jest efektywniejszy od innych rozwiązań postawionego problemu. Idealnym wynikiem naszych działań jest znalezienie optymalnego algorytmu, jednak nie zawsze jest to możliwe. Algorytm optymalny rozwiązuje problem w najkrótszym czasie, czyli przez wykonanie najmniejszej liczby operacji.

Przykład 1.2.

Wiemy, czym są lata przestępne. To wtedy rok ma 366 dni, a nie 365. Ale czy zdajemy sobie sprawę z tego, kiedy rok będzie przestępny? Spróbujmy skonstruować algorytm, który sprawdzi, czy podany rok, mający nastąpić w przyszłości, jest przestępny.

Zaczniemy od **określenia specyfikacji**. Dany mamy rok, który dopiero nastąpi. Wynikiem będzie informacja, czy podany rok jest przestępny.

Kolejnym krokiem jest **analiza problemu**. Musimy się dowiedzieć, jakie warunki powinien spełniać taki rok. Wiemy, że ma dopiero nastąpić, mamy więc do czynienia z kalendarzem gregoriańskim, który dokładnie określa te warunki.

Konstruujemy więc algorytm. Jeśli rok jest podzielny przez 4 i jednocześnie nie jest podzielny przez 100 lub jeśli rok jest podzielny przez 400, mamy rok przestępny, w przeciwnym razie rok nie jest przestępny.

Przejdźmy do **testowania algorytmu**. Na przykład rok 2020 jest podzielny przez 4 i niepodzielny przez 100 — jest więc przestępny. Kolejnym przykładem jest rok 2100 — podzielny przez 4 i podzielny przez 100, ponadto niepodzielny przez 400. Czy to jest rok przestępny? Podaj więcej przykładów.

Zadanie 1.1.

Pracownik pewnej firmy otrzymuje miesięczne wynagrodzenie zasadnicze w wysokości *k* złotych. Dodatkowo co dwa miesiące uzyskuje premię w wysokości 10% miesięcznej płacy podstawowej. Wynagrodzenie zasadnicze wzrasta co trzy miesiące o 5%. Pracownik otrzymuje wynagrodzenie wraz z premią zawsze ostatniego dnia danego miesiąca. Jaką płacę uzyska pracownik po sześciu miesiącach i po roku? Podaj specyfikację zadania, przeanalizuj problem i przedstaw sposób (czyli opis słowny algorytmu) obliczenia wynagrodzenia pracownika. Przetestuj skonstruowany algorytm dla przykładowych danych.

Temat 2. Sposoby przedstawiania algorytmów

Istnieje wiele sposobów przedstawiania algorytmów. Wybór odpowiedniej metody powinien zależeć od rozwiązywanego problemu. Do wykonywanego zadania dobierz taki sposób reprezentowania algorytmu, który najdokładniej i najczytelniej pokaże jego przebieg. Najważniejsze **metody przedstawiania algorytmów wykorzystywane w tym podręczniku** to:

- lista kroków,
- schemat blokowy,
- program w tekstowym języku programowania Python.

Do prezentowania algorytmów można stosować też inne metody, na przykład opis słowny, pseudokod, drzewo algorytmu, drzewo wyrażenia, inne języki programowania. Metody zastosowane w podręczniku poznaliście już w szkole podstawowej. Poniżej przypomnimy najważniejsze informacje na temat listy kroków i schematu blokowego. Język programowania Python zostanie wprowadzony w kolejnych rozdziałach od początku, ponieważ w szkole podstawowej mógł pojawić się inny tekstowy język programowania.

2.1. Lista kroków algorytmu

Jednym z najczęściej stosowanych sposobów prezentacji algorytmów jest lista kroków. Forma ta umożliwia dokładne przedstawienie kolejnych operacji realizowanych przez algorytm, z uwzględnieniem powtarzania działań, zakończenia algorytmu po spełnieniu określonych warunków czy zmiany kolejności wykonywanych operacji. Lista kroków jest listą numerowaną, w której każdy krok algorytmu ma przypisany numer. Kroki algorytmu zawarte w liście realizuje się kolejno, należy jednak uwzględniać zmianę kolejności spowodowaną wykonaniem polecenia przejścia do kroku o podanym numerze. Umożliwia to na przykład realizację powtarzania operacji w algorytmie.

W podręczniku w listach kroków zastosowano matematyczne symbole operacji i warunków.

Przykład 2.1.

Skonstruujmy algorytm w postaci listy kroków rozwiązujący **równanie liniowe** $ax + b = 0$, które rozwiązywaliście na lekcjach matematyki w szkole podstawowej. Zacznijmy od określenia **specyfikacji zadania**.

Specyfikacja:

Dane: dowolne liczby rzeczywiste: a, b.

Wynik: wartość rzeczywista pierwiastka równania liniowego: x lub komunikat informujący o braku rozwiązania bądź uzyskaniu nieskończenie wielu rozwiązań.

Kolejnym etapem będzie **przeprowadzenie analizy rozwiązywanego problemu**. Tworząc algorytm, należy rozpatrzyć wszystkie możliwe sytuacje. W przypadku równania liniowego rozwiązanie zadania zależy od wartości współczynników a i b, co przedstawia się następująco:

$a \neq 0$: $x = \frac{-b}{a}$,

$a = 0$ i $b = 0$: nieskończenie wiele rozwiązań (w tym przypadku rozwiązaniem jest każda liczba rzeczywista),

$a = 0$ i $b \neq 0$: równanie sprzeczne.

Po wykonaniu analizy zadania możemy przejść do konstruowania algorytmu.

Algorytm w postaci listy kroków:

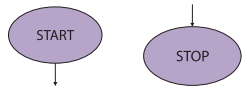
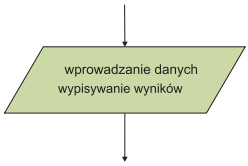
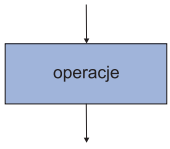
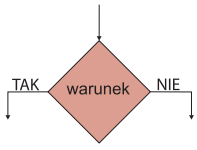
Krok 1. Jeśli $a \neq 0$, oblicz $x = \frac{-b}{a}$, wypisz wynik x i zakończ algorytm.

Krok 2. (W tym przypadku $a = 0$). Jeśli $b = 0$, wypisz komunikat „nieskończenie wiele rozwiązań", w przeciwnym razie wypisz komunikat „równanie sprzeczne". Zakończ algorytm.

2.2. Schemat blokowy algorytmu

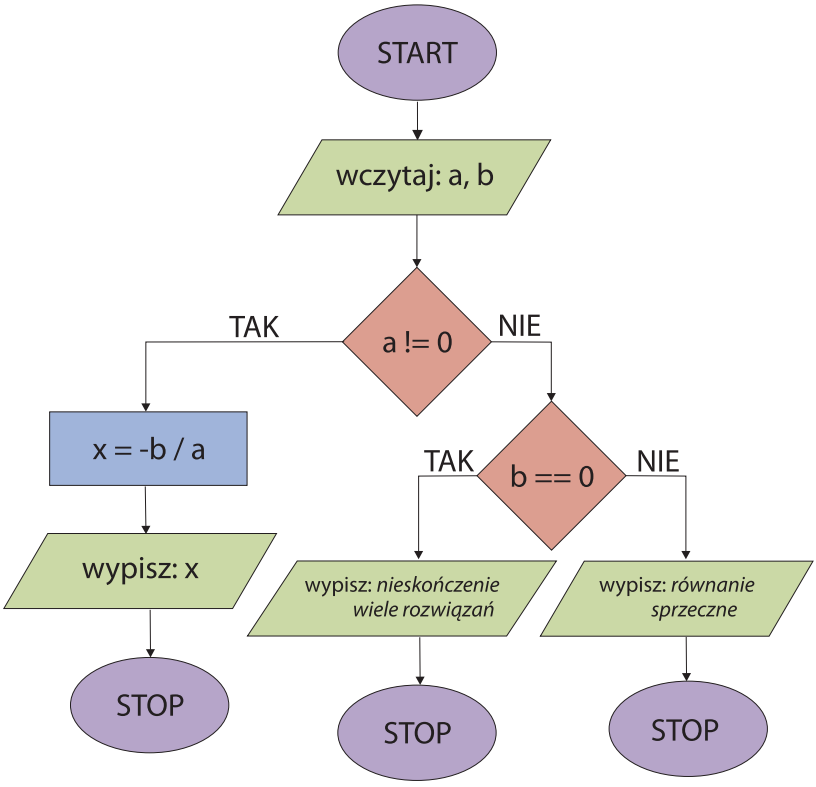
Schemat blokowy to graficzny sposób prezentacji algorytmu. Składa się z elementów przedstawionych w tabeli 2.1, zwanych blokami. Poszczególne części schematu należy łączyć, wykorzystując strzałki, które określają porządek wykonywania operacji.

Tabela 2.1. Elementy schematu blokowego

Element schematu blokowego	Opis elementu
	Początek i koniec algorytmu
	Blok wejścia i wyjścia (wprowadzanie danych lub wypisywanie wyników)
	Blok operacyjny (wykonywanie działań)
	Blok decyzyjny (konstruowanie warunków i sprawdzanie ich spełnienia)

Przykład 2.2.

Na rysunku 2.1 przedstawiono schemat blokowy algorytmu rozwiązującego równanie liniowe $ax + b = 0$, zapisanego w przykładzie 2.1 w postaci listy kroków.



Rysunek 2.1. Schemat blokowy algorytmu rozwiązującego równanie liniowe ax + b = 0

Przeanalizuj zaproponowane rozwiązanie widoczne na rysunku 2.1. Czy warunki są zgodne z przeprowadzoną wcześniej analizą?

W schemacie blokowym można stosować matematyczne symbole operacji i warunków, operatory danego języka programowania lub opisy słowne wykonywanych operacji. Zapis jednak musi być jednoznaczny i dokładnie określać wykonywane działania. Dla ułatwienia w podręczniku w schematach blokowych wykorzystywane są operatory języka Python, w którym będziesz zapisywać programy realizujące omawiane algorytmy.

2.3. Program w języku programowania wysokiego poziomu

Jednym z najważniejszych sposobów przedstawiania algorytmów jest program napisany w języku programowania wysokiego poziomu. Praktyczna implementacja algorytmu, jaką jest napisanie programu, ułatwia jego testowanie dla różnych danych wejściowych. Ponadto jest to sposób zrozumiały dla komputera.

Przykład 2.3.

Poniżej przedstawiono programy w różnych tekstowych językach programowania realizujące algorytm opisany w przykładach 2.1 i 2.2, rozwiązujący równanie liniowe $ax + b = 0$.

Program w języku Python:

```
def rownanie_liniowe(a, b):
    if a != 0:
        x = -b / a
        print("x = ", x)
    elif b == 0:
        print("nieskończenie wiele rozwiązań")
    else: print("równanie sprzeczne")

rownanie_liniowe(-2, 5)
```

Program w języku C++:

```
#include <iostream>
using namespace std;
int main()
{
  double a, b, x;
  cout << "podaj a, b: " << endl;
  cin >> a >> b;
  if (a != 0) { x = -b / a; cout << "x = " << x << endl; }
  else if (b == 0) cout << "nieskończenie wiele rozwiązań" << endl;
       else cout << "równanie sprzeczne" << endl;
  return 0;
}
```

Program w języku Pascal:

```
program rownanie_liniowe;
var a, b, x: real;
begin
  writeln('podaj a, b:');
  readln(a, b);
  if a <> 0 then begin x := -b / a; writeln('x = ', x) end
  else if b = 0 then writeln('nieskończenie wiele rozwiązań')
       else writeln('równanie sprzeczne')
end.
```

Przeanalizuj realizację algorytmu w różnych językach programowania. Który kod programu wydaje się najprostszy w zapisie? Który jest najkrótszy? Czy zauważasz instrukcje warunkowe?

Ćwiczenie 2.1.

Przetestuj działanie programów podanych w przykładzie 2.3 dla różnych danych wejściowych.

Zadanie 2.1.

Algorytm Euklidesa wyznacza największy wspólny dzielnik dwóch liczb naturalnych. Można go realizować z wykorzystaniem reszty z dzielenia lub za pomocą odejmowania. Algorytm ten pojawił się w szkole podstawowej, więc jest Ci już znany. Określ specyfikację tego algorytmu, wybierz metodę i przedstaw go za pomocą dowolnej notacji omówionej w tym temacie. Następnie przetestuj go dla przykładowych danych.

Temat 3. Klasyfikacja tekstowych języków programowania

Definicja

Języki programowania to specjalne języki przeznaczone do formułowania algorytmów w taki sposób, aby były zrozumiałe dla komputera. Możemy je podzielić na dwie grupy:

- **języki wewnętrzne** — każde słowo zapisywane jest w postaci ciągu 0 i 1 o ustalonej długości; słowo może być interpretowane jako liczba lub rozkaz;
- **języki zewnętrzne** — języki stworzone z myślą o zapisywaniu algorytmów przez człowieka w sposób zrozumiały dla komputera; dzielimy je na języki niskiego i wysokiego poziomu.

Konieczność stworzenia języków zewnętrznych została podyktowana tym, że zapisywanie algorytmów bezpośrednio w języku wewnętrznym z wykorzystaniem systemu binarnego jest uciążliwe. Dlatego programista zapisuje algorytm w postaci programu w wybranym języku zewnętrznym, który następnie jest tłumaczony na język wewnętrzny komputera.

Do **języków zewnętrznych niskiego poziomu** zaliczamy języki zorientowane maszynowo, czyli ściśle związane z określonym typem procesora, a więc również językiem wewnętrznym komputera. Należą do nich asemblery, których używa się zwykle do konstruowania „wstawek" w programach pisanych w językach wysokiego poziomu.

Języki zewnętrzne wysokiego poziomu służą do pisania programów przez programistów. Ich struktura jest czytelna dla człowieka i pozwala w prosty sposób zapisywać konstrukcje algorytmiczne pojawiające się w rozwiązywanym problemie.

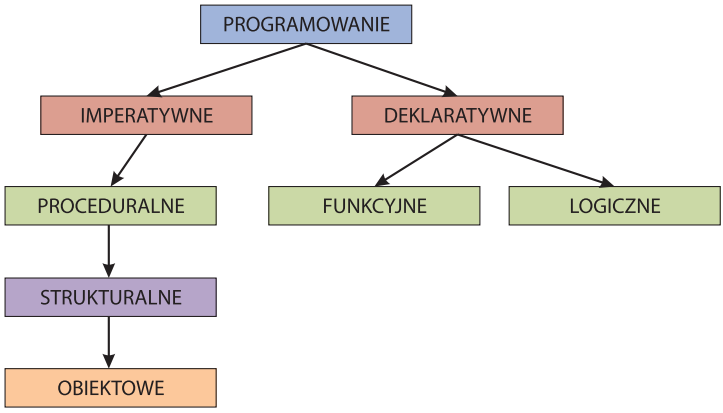
Definicja

Translacja to proces tłumaczenia programu z języka źródłowego (na przykład języka zewnętrznego wysokiego poziomu) na język wynikowy (na przykład język wewnętrzny) zrozumiały dla procesora.

Istnieją dwa typy programów realizujących tłumaczenie:

- **kompilatory** — najpierw cały program napisany w języku programowania jest tłumaczony na język wewnętrzny, a następnie wykonywany;
- **interpretery** — każde polecenie jest kolejno tłumaczone na język wewnętrzny i wykonywane przez komputer.

Na rysunku 3.1 przedstawiono **podstawowy podział języków programowania** na dwie grupy: języki imperatywne i deklaratywne.



Rysunek 3.1. Podstawowy podział języków programowania

Języki imperatywne charakteryzują się tym, że:

- używają rozkazów określających czynności, które komputer ma wykonywać;
- program jest listą rozkazów do wykonania, stąd nazwa „imperatywne", czyli rozkazowe;
- mówią komputerowi, jak ma osiągnąć wynik, choć nie określają, jaki to ma być wynik.

W programowaniu imperatywnym program jest listą instrukcji, a instrukcje wykonywane są kolejno z możliwością wielokrotnego powtarzania pewnych czynności zapisanych raz, czyli wykonywania pętli (iteracji).

Programowanie proceduralne to najstarszy typ programowania imperatywnego. W kodzie programu wydzielone są fragmenty zwane podprogramami (procedury, funkcje, metody, operacje), które mogą być wielokrotnie uruchamiane. Ten typ programowania przyczynił się do powstania techniki programowania *bottom-up*, polegającej na pisaniu kodu od małych fragmentów do coraz większych i w końcu do stworzenia całego programu. Przykładami języków proceduralnych są asemblery, Basic, Fortran, Pascal, C, C++, Object Pascal, Lisp, Logo, Python, Ruby. Te języki należą do grupy języków imperatywnych. Zauważ, że prawie wszystkie wymieniane języki można zaliczyć do różnych grup. Są to języki ogólnego przeznaczenia, które mają szerokie zastosowanie, przez co rozwijają się w różnych kierunkach.

Przykład 3.1.

Przeanalizuj przykładowy program napisany w języku proceduralnym Pascal.

Program w języku Pascal:

```
program suma;
var i, n, s: integer;
begin
  read(n);
  suma := 0;
  for i := 1 to n do
    s := s + i;
  write(s)
end.
```

Powyższy program oblicza sumę kolejnych liczb naturalnych 1, 2, …, *n*, dla *n* podanego z klawiatury, a następnie wypisuje obliczony wynik.

Programowanie strukturalne to rozwinięcie programowania proceduralnego. Korzysta z możliwości zapisywania każdego algorytmu za pomocą podstawowych struktur, takich jak sekwencja, instrukcje warunkowe, pętle, funkcje. Ten typ programowania umożliwia stosowanie techniki *top-down*, która jest odwróceniem techniki *bottom-up*. Polega ona na dzieleniu całego zadania na mniejsze części zgodnie z przewidywaną strukturą pisanego programu oraz na wypełnianiu tej struktury podstawowymi poleceniami. Przykładami języków strukturalnych są: Pascal, C, C++, Object Pascal, Python, Ruby.

Programowanie obiektowe to rozwinięcie programowania strukturalnego. Programy definiuje się tutaj za pomocą obiektów — elementów łączących stan (czyli dane nazywane polami) i zachowanie (czyli metody). Program napisany obiektowo wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań. Uważa się, że ten typ programowania dobrze odzwierciedla sposób, w jaki ludzie myślą o świecie — widzimy obiekty i ich cechy (czyli pola), oraz operacje, które na nich możemy wykonywać. Jeśli obiektem jest szkoła, to polami są uczniowie i nauczyciele, a wszystko, co robimy w szkole, to metody. Do języków obiektowych zaliczyć można języki: C++, Object Pascal, C#, Java, Python, Ruby.

Przykład 3.2.

Przykładowy program w języku obiektowym Java.

Program w języku Java:

```
public class Hello {
  public static void main(String[] args) {
    System.out.println("Witaj");
  }
}
```

Ten program wypisuje na ekranie komunikat „Witaj”.

Języki deklaratywne to języki, w których programista podaje komputerowi pewne zależności oraz cele, jakie program ma osiągnąć. Opisują one, co ma zostać osiągnięte, ale nie podają, w jaki sposób.

Programowanie funkcyjne to odmiana programowania deklaratywnego. Program jest tutaj złożoną funkcją, która na podstawie danych wejściowych wylicza pewien wynik. To programowanie różni się od tych wcześniej omówionych przede wszystkim brakiem zmiennych i instrukcji pętli. Konstruowanie programów to składanie funkcji, zazwyczaj z wykorzystaniem rekurencji, czyli uruchamiania funkcji wewnątrz niej samej. Przykładami języków funkcyjnych są: Lisp, Logo, Python, Ruby.

Programowanie logiczne to kolejny rodzaj programowania deklaratywnego. Program jest tutaj zbiorem zależności (przesłanki) i pewnych stwierdzeń (cel), a wykonanie programu to próba udowodnienia celu na podstawie podanych przesłanek. Nie piszemy poleceń, a jedynie opisujemy, co wiemy i co chcemy uzyskać. Przykładem języka logicznego jest Prolog.

Języki te wykorzystywane są przede wszystkim w zakresie sztucznej inteligencji, systemów eksperckich i innych pokrewnych dziedzin.

Przykład 3.3.

Poniżej podano przykładowy program w języku logicznym Prolog.

Program w języku Prolog:

```
ojciec(Jan, Adam).
ojciec(Adam, Krzysztof).
ojciec(Adam, Piotr).
dziadek(X, Z) :- ojciec(X, Y), ojciec(Y, Z).
?- dziadek(X, Krzysztof).
```

Przeanalizujmy kod tego programu:

- Jan jest ojcem Adama.
- Adam jest ojcem Krzysztofa.
- Adam jest ojcem Piotra.
- Jeśli X jest dziadkiem Z, to X jest ojcem Y i Y jest ojcem Z.
- X jest dziadkiem Krzysztofa. Kim jest X?

Spróbuj na podstawie podanych informacji odpowiedzieć na to pytanie i podać, kim jest X.

Zadanie 3.1.

Odpowiedz na podane pytania:

a) Jaka jest różnica między kompilacją a interpretacją kodu programu?

b) Czym różni się programowanie imperatywne od deklaratywnego?

c) Na czym polega różnica między technikami programowania *bottom-up* i *top-down*?

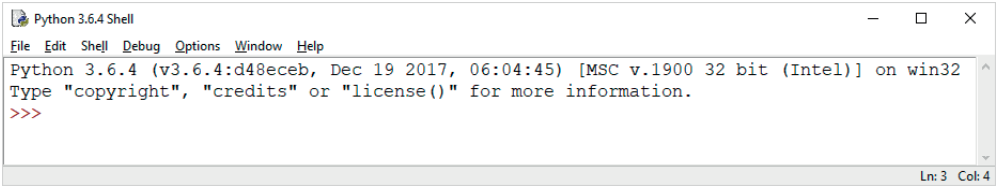
Temat 4. Python od pierwszych kroków

Python to język programowania wysokiego poziomu. Ma on przejrzystą i zwięzłą składnię. Programy pisane są w plikach tekstowych z rozszerzeniem *.py*. W języku Python istotna jest wielkość liter.

W kodach programów **wykorzystuje się wcięcia do wydzielania bloków kodu**. Przyjęło się stosowanie czterech spacji, ale nie jest to konieczne. Jednak musimy być konsekwentni, co oznacza, że jeśli pierwsze wcięcie w funkcji miało trzy spacje, to kolejne wcięcia także muszą mieć trzy spacje. Linia bez wcięcia będzie się znajdować poza blokiem.

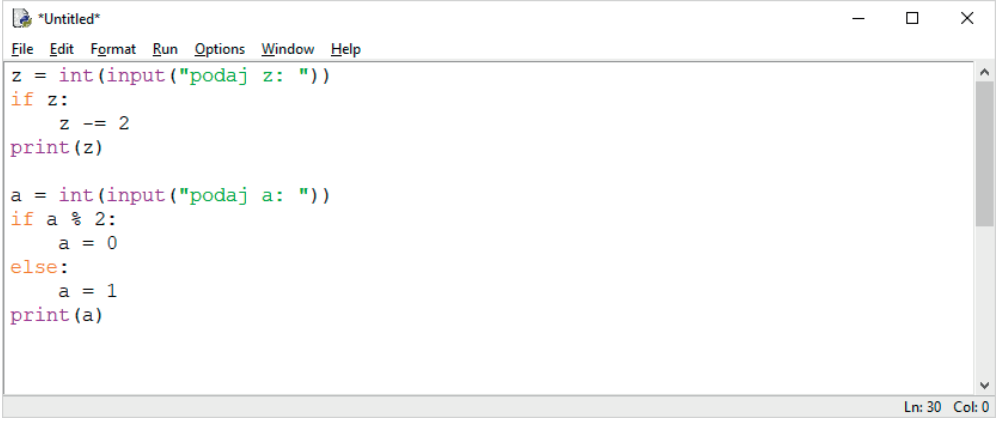
4.1. Środowisko programowe Pythona

IDLE to zintegrowane środowisko programistyczne tego języka dołączone do interpretera Pythona, które można znaleźć na stronie *https://www.python.org/*. Oprogramowanie jest darmowe, a interpretery są dostępne na wiele systemów operacyjnych. Zintegrowane środowisko programowe *IDLE* to zestaw narzędzi, które ułatwiają pisanie programów w Pythonie. Po uruchomieniu *IDLE* zgłasza się **tryb interaktywny**, przedstawiony na rysunku 4.1.



Rysunek 4.1. Tryb interaktywny środowiska programowania IDLE

Aby przejść do edycji nowego programu, należy z menu *File* wybrać polecenie *New File*. Otworzy się okno edytora przeznaczone do pisania programów — **tryb skryptowy** — co widać na rysunku 4.2.



Rysunek 4.2. Tryb skryptowy środowiska programowania IDLE

Istnieją też inne środowiska, w których można pisać programy w tym języku, na przykład *PyCharm*, edytor *Geany*, edytor *Mu*. Można również korzystać ze środowisk udostępnianych online. Być może na lekcjach informatyki w szkole podstawowej korzystaliście z różnych środowisk programowania w języku Python.

Aby rozpocząć pisanie programu w określonym języku programowania, powinno się poznać jego podstawowe polecenia. Bez znajomości „słów" nie jesteśmy w stanie porozumiewać się w żadnym języku. Zapoznaj się z podanymi poniżej informacjami i przeanalizuj przykłady. Rozpocznij pisanie swoich pierwszych programów w języku Python w środowisku wybranym przez nauczyciela.

4.2. Operacje wejścia-wyjścia

W tabeli 4.1 podano **podstawowe operacje wejścia-wyjścia języka Python**.

Tabela 4.1. Podstawowe operacje wejścia-wyjścia w języku Python

Polecenie	Opis polecenia
`print("tekst", zmienna)`	Wypisanie komunikatu na ekranie
`a = float(input("podaj liczbę rzeczywistą: "))` `b = int(input("podaj liczbę całkowitą: "))` `c = input("podaj tekst: ")`	Wprowadzanie wartości zmiennych z klawiatury (zmienne są tworzone przez przypisanie wartości)

Przykład 4.1.

Przykład pokazuje zastosowanie operacji wejścia-wyjścia języka Python. Po wprowadzeniu daty urodzenia, kolejno dnia, miesiąca i roku wypisywana jest podana data w odpowiednim formacie.

```
d = int(input("podaj dzień: "))
m = input("podaj miesiąc: ")
r = int(input("podaj rok: "))
print(d, m, r, "r.")
```

Dla przykładowych danych program powinien działać następująco:

podaj dzień: 23

podaj miesiąc: marca

podaj rok: 2008

23 marca 2008 r.

Ćwiczenie 4.1.

Napisz program w języku Python, w którym wypiszesz na ekranie komunikat: *MÓJ PIERWSZY PROGRAM W JĘZYKU PYTHON*.

Ćwiczenie 4.2.

Napisz program w języku Python, w którym przypiszesz wartości zmiennym różnego typu poprzez wpisanie ich z klawiatury, a następnie wypiszesz wartości tych zmiennych na ekranie. Przykładowy wynik działania programu powinien być taki:

podaj liczbę rzeczywistą: 25.14

podaj liczbę całkowitą: 378

podaj tekst: Python

liczba rzeczywista = 25.14

liczba całkowita = 378

tekst = Python

Zwróć uwagę na zapis liczby rzeczywistej.

4.3. Komentarze

Komentarz to tekst zawarty w kodzie programu, który nie jest analizowany przez interpreter. Wykorzystywany jest on do komentowania programu, w którym został umieszczony. Aby tekst został potraktowany jako komentarz, oznacza się go odpowiednimi znakami. Istnieją dwa rodzaje komentarzy, co zostało przedstawione w tabeli 4.2. Stosuj komentarze zawsze, gdy chcesz dopisać swoją uwagę, zrobić notatkę lub wyłączyć z działania programu fragment kodu.

Tabela 4.2. Komentarze w języku Python

Polecenie	Opis polecenia
`# komentarz`	Jednowierszowy komentarz
`""" komentarz """` `''' komentarz '''`	Wielowierszowe komentarze ograniczone przez " lub '

4.4. Operatory arytmetyczne

Operatory arytmetyczne (tabela 4.3) to operatory dwuargumentowe realizujące operacje arytmetyczne.

Tabela 4.3. Zestawienie operatorów arytmetycznych

Polecenie	Opis polecenia
+	Dodawanie
–	Odejmowanie
*	Mnożenie
/	Dzielenie

Polecenie	Opis polecenia
//	Dzielenie całkowite
%	Reszta z dzielenia
**	Potęgowanie

Przykład 4.2.

Poniżej podano przykłady zastosowania i poprawnego zapisu operacji arytmetycznych:

```
a = 5 + 3 - b
b = a / 4
c = a * (b - 1)
d = 2 ** a
```

W wyrażeniach możemy stosować nawiasy.

Ćwiczenie 4.3.

Napisz program, w którym wykonasz działania podane w przykładzie 4.2 dla danych wprowadzanych z klawiatury. Przykładowy program powinien mieć taką postać:

```
b = int(input("podaj b: "))
a = 5 + 3 - b
print("a = ", a)
```

Po uruchomieniu programu możesz go przetestować:

podaj b: 7

a = 1

Zadanie 4.1.

Napisz programy, w których wykonasz działania podane poniżej dla danych wprowadzanych z klawiatury:

a) $a = b^3 + 5$, b) $a = \frac{4-b}{5}$, c) $a = \frac{2^3 + b}{3^2}$.

4.5. Złożone operatory przypisania

Złożone operatory przypisania stosuje się do zapisywania wyrażeń `X = X · Y` w postaci `X ·= Y`, gdzie `·` to operator dwuargumentowy. Do najważniejszych z nich należą:

```
+=  -=  *=  /=  //=  %=  **=
```

Przykład 4.3.

W tabeli 4.4 podano przykłady zastosowania złożonych operatorów przypisania w konstruowaniu wyrażeń.

Tabela 4.4. Przykłady zastosowania złożonych operatorów przypisania

Wyrażenie	Zapis wyrażenia bez złożonego operatora przypisania
`a *= 4 * b - 1`	`a = a * (4 * b - 1)`
`m %= 3 - n / 2`	`m = m % (3 - n / 2)`
`b -= 3 + d * 4`	`b = b - (3 + d * 4)`

Zwróć uwagę na kolejność wykonywanych działań.

Ćwiczenie 4.4.

Napisz program, w którym wykonasz działania podane w przykładzie 4.3 dla danych wprowadzanych z klawiatury. Przykładowy program powinien wyglądać tak:

```
a = int(input("podaj a: "))
b = int(input("podaj b: "))
a *= 4 * b - 1
print("a = ", a)
```

Po uruchomieniu programu możesz go przetestować:

podaj a: 2

podaj b: 3

a = 22

Zadanie 4.2.

Napisz programy, w których wykonasz działania podane poniżej z wykorzystaniem złożonych operatorów przypisania dla danych wprowadzanych z klawiatury:

a) $a = a \cdot \left(3^3 + b\right)$, b) $k = k - \left(i \cdot 5 + 2\right)$, c) $w = \frac{w}{4 + z^2}$.

Temat 5. Algorytmy liniowe

Definicja

Algorytm liniowy (zwany również sekwencyjnym) to rodzaj algorytmu, w którym wszystkie instrukcje wykonywane są kolejno, bez konieczności rozpatrywania warunków.

Kolejność wykonywanych operacji jest więc w algorytmach tego typu zawsze taka sama i nie zależy od wczytywanych danych wejściowych. Schemat blokowy jest tutaj strukturą liniową, bez rozgałęzień. Program zawiera wyłącznie operacje wejścia-wyjścia oraz instrukcje przypisania.

Przeanalizuj podane poniżej przykłady i spróbuj samodzielnie wykonać zadania.

Przykład 5.1.

Dana jest liczba naturalna n większa od 0. Obliczmy sumę ciągu n kolejnych liczb naturalnych: 1, 2, …, n. Algorytm zrealizujemy w postaci listy kroków, schematu blokowego (rysunek 5.1) oraz programu w języku Python.

Specyfikacja:

Dane: liczba naturalna: $n > 0$.

Wynik: suma n kolejnych liczb naturalnych większych od 0: s.

Analiza problemu:

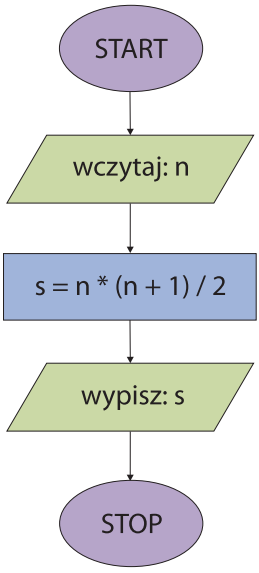
Zadanie można wykonać, sumując elementy ciągu lub korzystając ze wzoru na sumę częściową ciągu arytmetycznego. Stosując drugą metodę, otrzymujemy następujące wyrażenie:

$$1+2+\ldots+n=\frac{n(n+1)}{2}.$$

Lista kroków:

Krok 1. Przypisz $s=\frac{n(n+1)}{2}$.

Krok 2. Wypisz wynik *s*. Zakończ algorytm.



Rysunek 5.1. Schemat blokowy przykładowego algorytmu liniowego

Program w języku Python:

```
n = int(input("podaj n = "))
s = n * (n + 1) // 2
print(s)
```

Przetestuj działanie programu dla różnych danych.

Ćwiczenie 5.1.

Wśród algorytmów, które dotychczas analizowaliśmy we wcześniejszych przykładach, pojawiły się już algorytmy liniowe. Kolejne polecenia w tych algorytmach realizowane są sekwencyjnie, jedno po drugim, bez żadnych warunków. Przejrzyj poprzednie tematy i znajdź algorytmy, które nie są liniowe. Czym różnią się od liniowych?

Zadanie 5.1.

Podaj specyfikację zadania i skonstruuj algorytm w postaci listy kroków oraz program obliczający pole i obwód wybranej figury geometrycznej, na przykład prostokąta. Wymiary figury wprowadź za pomocą klawiatury. Przetestuj program dla różnych danych.

Zadanie 5.2.

Podaj specyfikację zadania i napisz program wykonujący podstawowe działania arytmetyczne na liczbach wprowadzonych z klawiatury. Przetestuj program dla różnych danych. Poniżej pokazano przykładowy wynik działania programu:

podaj a = 3

podaj b = 2

3 + 2 = 5

3 – 2 = 1

*3 * 2 = 6*

3 / 2 = 1.5

3 // 2 = 1

3 % 2 = 1

*3 ** 2 = 9*

Zadanie 5.3.

Podaj specyfikację zadania i skonstruuj algorytm w postaci schematu blokowego oraz programu, obliczający średnią arytmetyczną trzech liczb całkowitych *a*, *b* i *c* wprowadzonych z klawiatury.

Temat 6. Funkcje matematyczne w języku Python

W języku Python mamy dostęp do wielu **funkcji matematycznych**, które są niezbędne przy wykonywaniu niektórych obliczeń. Zawarte są one w bibliotece `math`, którą należy zadeklarować.

W tabeli 6.1 przedstawiono dostęp do tej biblioteki i zestawienie podstawowych funkcji matematycznych przez nią oferowanych.

Tabela 6.1. Zestawienie podstawowych funkcji matematycznych biblioteki math

Polecenie	Opis polecenia
`from math import *`	Dostęp do biblioteki matematycznej
`a = sqrt(49)`	Zastosowanie funkcji `sqrt()`, obliczającej pierwiastek kwadratowy, wynik: *a = 7*
`b = fabs(-8)`	Zastosowanie funkcji `fabs()`, obliczającej wartość bezwzględną, wynik: *b = 8*
`a = sin(1)` `b = cos(1)` `c = tan(1)`	Zastosowanie funkcji trygonometrycznych `sin()`, `cos()` i `tan()` (argumenty podajemy w radianach), wyniki: *a = 0.8414709848078965* *b = 0.5403023058681398* *c = 1.5574077246549023*

Przykład 6.1.

Obliczmy wartość podanego wyrażenia w języku Python:

$$w = \sqrt{\frac{7}{a^3 + \cos(b)}}.$$

Wartości zmiennych *a* i *b* wprowadzamy z klawiatury, a następnie wykonujemy obliczenia.

Przyjrzyj się, w jaki sposób zastosowano funkcje matematyczne języka Python w programie.

```
from math import *
a = float(input("podaj liczbę a = "))
b = float(input("podaj liczbę b = "))
w = sqrt(7 / (a ** 3 + cos(b)))
print("wynik = ", w)
```

Przykładowe wyniki:

podaj liczbę a = 4

podaj liczbę b = 5

wynik = 0.3299884314684228

Przetestuj ten program dla różnych danych.

Zadanie 6.1.

Średnią geometryczną dwóch liczb nieujemnych a i b nazywamy pierwiastek kwadratowy iloczynu tych liczb. Określ specyfikację zadania i skonstruuj algorytm w postaci programu, obliczający średnią geometryczną dwóch liczb wprowadzonych z klawiatury.

Zadanie 6.2.

Poniżej podano wyrażenia, których wartość należy obliczyć, pisząc program w języku Python. Znasz już funkcje matematyczne w tym języku (tabela 6.1). Zastosuj je i oblicz wartości podanych wyrażeń dla danych wprowadzanych z klawiatury.

a) $w = a^3 + \cos(b) \cdot \sqrt{a+b}$,

b) $w = |a-b| + \sin(a) \cdot \sqrt{b}$,

c) $w = \sqrt{\dfrac{3+\sqrt{a \cdot b}}{|b^2 - 20|}}$,

d) $w = \sin\left(\dfrac{(a+b)^4}{\sqrt{11} + \sin(b)}\right)$,

e) $w = \left(\dfrac{\cos(a+1)}{|\sqrt{5} - b|}\right)^3$.

Temat 7. Algorytmy warunkowe

Definicja

Algorytmy zawierające warunki, od których spełnienia zależy kolejność wykonywanych działań, nazywane są **algorytmami z warunkami**.

Przykładem takiego algorytmu jest omówiony w temacie 2. algorytm rozwiązujący równanie liniowe $ax + b = 0$, przedstawiony w postaci listy kroków, schematu blokowego (rysunek 2.1) oraz programów w językach wysokiego poziomu: Python, C++ i Pascal.

Schemat blokowy algorytmu z warunkami ma kształt drzewa, a w programach wykorzystywane są dodatkowo instrukcje warunkowe.

Skorowidz

G

H

I

J

K

R

S

Ś

T

U

W

Z

Ź

Technologia informacyjna i informatyka to dziedziny, których wykorzystanie i dostępność stale rosną, a tempo zachodzących w nich zmian jest nieporównywalne z rozwojem innych dyscyplin. Umiejętność biegłego posługiwania się komputerem i urządzeniami peryferyjnymi oraz znajomość obsługi pakietów biurowych są obowiązkowe na rynku pracy. Wkrótce każdy z nas będzie musiał się również legitymować wiedzą z zakresu podstaw programowania, tworzenia grafiki czy zagadnień związanych z prawem w sieci.

Informatyka w szkole ponadpodstawowej na poziomie podstawowym:

- jest przedmiotem obowiązkowym prowadzonym we wszystkich klasach
- nie jest przedmiotem maturalnym
- stanowi poziom podstawowy dla przedmiotu informatyka na poziomie rozszerzonym

Dzięki pracy z książką ***Informatyka Europejczyka. Podręcznik dla szkół ponadpodstawowych. Zakres podstawowy. Część 1*** nauczysz się szybko i sprawnie rozwiązywać problemy z użyciem komputera. Poznasz przyjazny i łatwy do opanowania język Python, co będzie świetnym wprowadzeniem do programowania. Opanujesz wiedzę z zakresu ochrony danych osobowych oraz prawa autorskiego — w efekcie będziesz bardziej świadomie i bezpiecznie korzystać z internetu. Dowiesz się, jak tworzyć atrakcyjne grafiki i prezentacje multimedialne, co z pewnością przyda Ci się przy przygotowywaniu referatów z historii, geografii i biologii. Znajomość zaawansowanych możliwości edytorów tekstu może okazać się zbawienna, gdy zechcesz napisać na przykład dłuższe wypracowanie i podczas przygotowań do matury z języka polskiego oraz potem — na studiach.

W skład kompletnego pakietu edukacyjnego dla szkoły ponadpodstawowej wchodzą także podręczniki:

Podręczniki z serii *Informatyka Europejczyka* ułatwią uczniom zdobywanie wiedzy i umiejętności podczas wykonywania ćwiczeń praktycznych, a nauczycielom — przekazywanie nowego materiału w interesujący i niebanalny sposób.

Wciśnij Enter i do dzieła!

http://edukacja.helion.pl

Helion EDUKACJA
zamówienia telefoniczne
0 801 339900
0 601 339900
księgarnia internetowa
http://helion.pl
Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl
Informatyka w najlepszym wydaniu
ISBN 978-83-283-5704-4
9 788328 357044