owoce programowania

Helion

>>> PROJEKTOWANIE OPROGRAMOWANIA DLA ZUPEŁNIE POCZĄTKUJĄCYCH

WYDANIE V

TONY GADDIS

Tytuł oryginału: Starting Out with Programming Logic and Design (5th Edition)

Tłumaczenie: Wojciech Moch z wykorzystaniem fragmentów książki „Projektowanie oprogramowania dla zupełnie początkujących. Owoce programowania. Wydanie IV" w tłumaczeniu Krzysztofa Braunera

ISBN: 978-83-283-5565-1

Authorized translation from the English language edition, entitled: STARTING OUT WITH PROGRAMMING LOGIC AND DESIGN, 5th Edition, by GADDIS TONY, published by Pearson Education, Inc, publishing as Pearson.

Helion SA
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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem: *ftp://ftp.helion.pl/przyklady/pkpro5.zip*

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/pkpro5
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 1

Wstępne informacje na temat komputerów i programowania

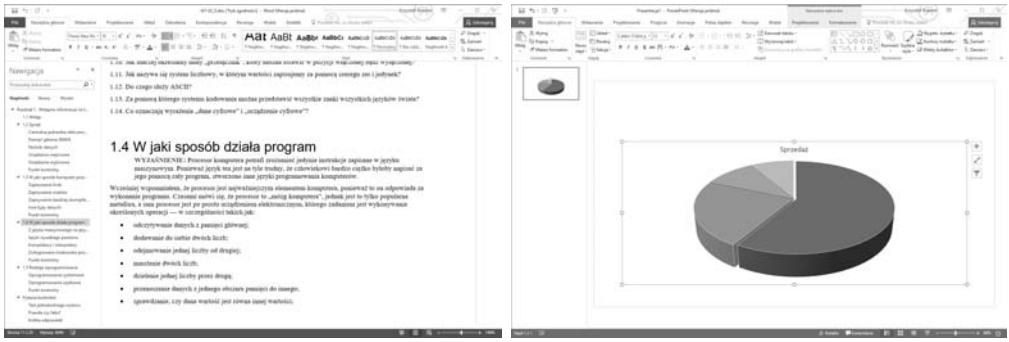
TEMATYKA

1.1 Wstęp

Pomyśl przez chwilę, na jak wiele sposobów ludzie korzystają z komputerów. Uczniowie w szkole piszą na nich wypracowania, wyszukują artykuły, wysyłają wiadomości e-mail i biorą udział w zajęciach online. W pracy natomiast za pomocą komputerów analizuje się dane, tworzy prezentacje, przeprowadza transakcje biznesowe, nawiązuje kontakty z klientami i współpracownikami, steruje maszynami w fabrykach, a także wykonuje się wiele innych czynności. W domu wykorzystujemy komputery do takich czynności jak płacenie rachunków, zakupy w sieci, kontakty z przyjaciółmi i rodziną oraz granie w gry. Zwróć uwagę, że urządzenia takie jak na przykład smartfony, tablety, odtwarzacze MP3 czy systemy nawigacji samochodowej są także komputerami. W życiu codziennym używamy komputerów niemal na każdym kroku.

Komputery potrafią wykonywać tak wiele zadań dlatego, że można je w dowolny sposób zaprogramować. Oznacza to, że komputery nie zostały stworzone, aby wykonywać jedno, określone zadanie, ale po to, aby wykonywać zadanie, jakie wskaże im określony program. **Program** składa się z szeregu instrukcji, które komputer musi uruchomić, aby wykonać określone zadanie. Na rysunku 1.1 przedstawione są dwa popularne programy: Microsoft Word i Microsoft PowerPoint.

Programy to inaczej **oprogramowanie**. Z punktu widzenia komputera oprogramowanie jest rzeczą kluczową — bez niego nie byłby w stanie wykonać żadnej operacji. Programy, dzięki którym nasze komputery są dla nas tak przydatne, tworzą twórcy oprogramowania, czyli **programiści**. Osoby te posiadają wiedzę dotyczącą projektowania, tworzenia i testowania programów komputerowych. Bycie programistą jest



Rysunek 1.1. Popularne programy komputerowe (zdjęcie dzięki uprzejmości Microsoft Corporation)

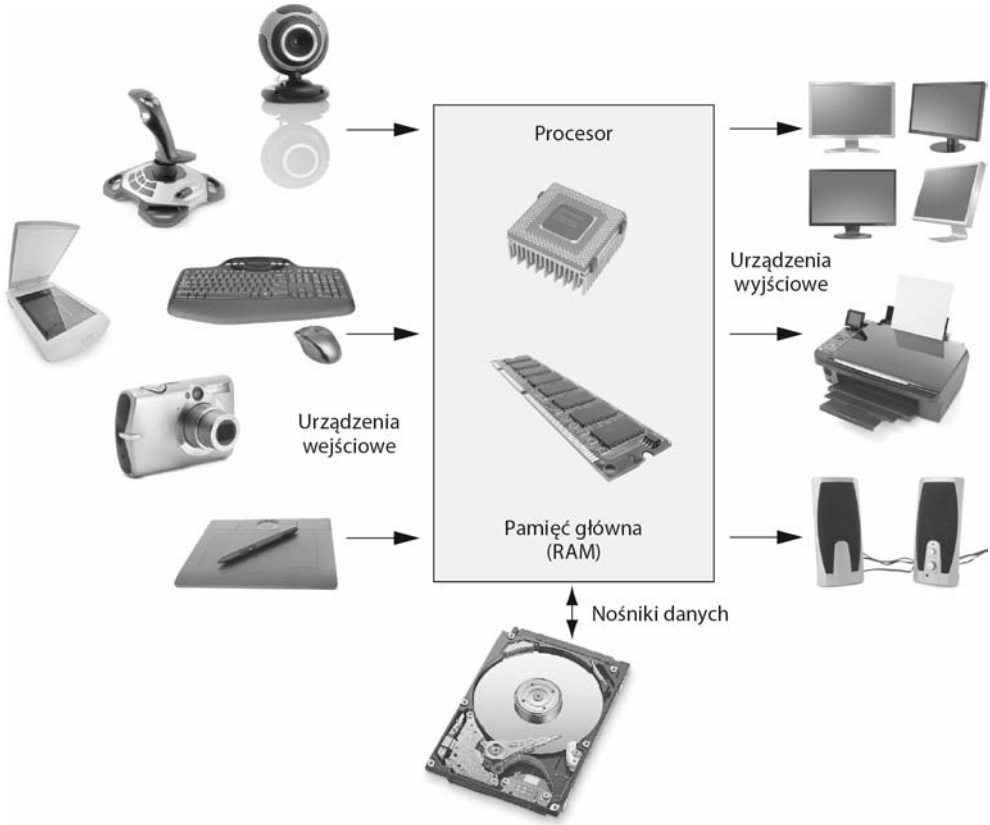
ekscytującym i satysfakcjonującym zajęciem. W dzisiejszych czasach programistów można znaleźć w bardzo wielu branżach: biznesie, medycynie, organach administracji publicznej, organach ścigania, rolnictwie, szkolnictwie, przemyśle rozrywkowym — niemal w każdej dziedzinie.

Dzięki tej książce poznasz podstawowe zagadnienia związane z programowaniem komputerów. Zanim jednak przejdziemy do omawiania tych zagadnień, musisz zrozumieć kilka prostych rzeczy dotyczących komputerów i ich działania. W tym rozdziale zdobędziesz wystarczającą wiedzę, która umożliwi Ci dalsze pogłębianie tematyki związanej z komputerami. Na początku omówię fizyczne elementy, z których składa się komputer. W kolejnym kroku przyjrzysz się temu, w jaki sposób komputer przechowuje dane i uruchamia programy. Na koniec rozdziału przybliżę główne typy oprogramowania komputerowego.

1.2 Sprzęt

WYJAŚNIENIE: **Sprzętem nazywamy wszystkie urządzenia fizyczne, z których zbudowany jest komputer. W zdecydowanej większości przypadków komputery składają się z bardzo podobnych urządzeń.**

Przez słowo **sprzęt** (ang. *hardware*) rozumiemy wszystkie urządzenia fizyczne (komponenty), z których zbudowany jest komputer. Komputer nie jest więc pojedynczym urządzeniem, lecz systemem składającym się z wielu urządzeń, które ze sobą współpracują. Podobnie jak w przypadku poszczególnych instrumentów w orkiestrze, każde urządzenie odgrywa w komputerze określoną rolę. Jeśli kiedyś kupowałeś komputer, zauważyłeś zapewne, że jest on opisany za pomocą listy komponentów takich jak procesor, pamięć, napędy, monitor, karta graficzna itp. Jeżeli nie masz dostatecznej wiedzy na temat komputerów lub nie znasz kogoś, kto ją posiada, zrozumienie takiego opisu może okazać się kłopotliwe. Na rysunku 1.2 przedstawiłem główne elementy, z jakich składa się typowy komputer:



Rysunek 1.2. Typowe elementy, z których składa się komputer (wszystkie zdjęcia © Shutterstock)

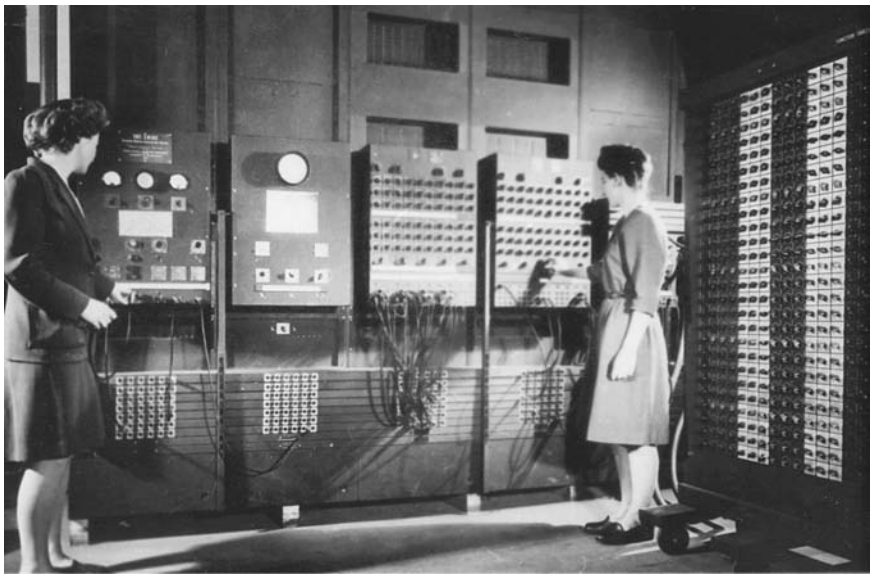
- centralna jednostka obliczeniowa (CPU),
- pamięć główna,
- nośniki danych,
- urządzenia wejściowe,
- urządzenia wyjściowe.

Przyjrzyjmy się teraz bliżej każdemu z wymienionych elementów.

Centralna jednostka obliczeniowa (CPU)

Kiedy komputer wykonuje wskazane przez program działania, mówimy, że **uruchomił** lub **wykonuje** dany program. To właśnie **centralna jednostka obliczeniowa** (ang. *central processing unit* — CPU) jest elementem komputera odpowiedzialnym za wykonywanie programu. Zamiast CPU przeważnie używa się nazwy **procesor**. Jest on najważniejszym elementem komputera, ponieważ bez niego komputer nie mógłby wykonać żadnego programu.

Pierwsze komputery były wyposażone w ogromne procesory, zbudowane z elektrycznych i mechanicznych elementów, takich jak lampy elektronowe i przełączniki. Na rysunku 1.3 widoczny jest właśnie taki komputer — dwie kobiety przedstawione na zdjęciu obsługują historyczny komputer **ENIAC**. Zbudowany w 1945 roku i używany przez armię Stanów Zjednoczonych do wyliczania tablic balistycznych, ENIAC uważany jest za pierwszy na świecie programowalny komputer elektroniczny. Maszyna ta (będąca tak naprawdę jednym wielkim procesorem) mierzyła 2,4 metra wysokości i 30 metrów długości, a ważyła 30 ton.



Rysunek 1.3. Komputer ENIAC (zdjęcie dzięki uprzejmości US ARMY Center of Military History)

Obecnie CPU to małe chipy zwane także **mikroprocesorami**. Na rysunku 1.4 widoczny jest inżynier trzymający współczesny mikroprocesor. Poza tym, że dzisiejsze mikroprocesory są znacznie mniejsze od swoich wczesnych elektromechanicznych odpowiedników, mają również znacznie większą moc obliczeniową.

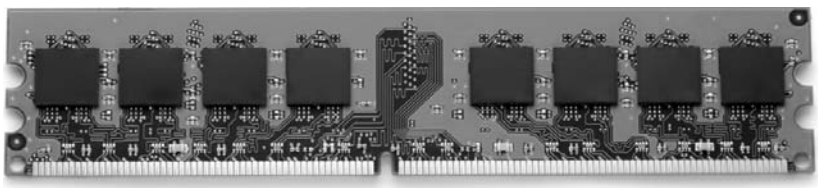
Pamięć główna (RAM)

Wyobraź sobie, że **pamięć główna** (ang. *main memory*) to przestrzeń robocza komputera. Jest to miejsce, w którym komputer przechowuje uruchomiony program, jak również dane, na których ten program pracuje. Przykładowo załóżmy, że piszesz wypracowanie w edytorze tekstowym — w takim przypadku zarówno program w postaci edytora tekstowego, jak i samo wypracowanie zostaną umieszczone w pamięci głównej komputera.

Pamięć główną nazywa się także **pamięcią o dostępie swobodnym** (ang. *random-access memory* — RAM). Nazwa ta wynika z tego, że procesor musi mieć szybki dostęp do każdej informacji zapisanej w dowolnym miejscu pamięci. RAM jest pamięcią

Rysunek 1.4. Inżynier trzymający współczesny mikroprocesor (zdjęcie dzięki uprzejmości Chris Ryan/OJO Images/Getty Images)

ulotną i wykorzystuje się ją jako tymczasowe miejsce zapisywania informacji — tylko na czas działania programu. Kiedy komputer się wyłączy, wszystkie dane zostaną wymazane z pamięci głównej. Pamięć RAM ma postać płytki z chipami — takiej, jaką widać na rysunku 1.5.



Rysunek 1.5. Chipy pamięci (zdjęcie © Garsya/Shutterstock)

UWAGA: Inny rodzaj pamięci, która jest zbudowana z chipów, to **pamięć tylko do odczytu** (ang. *read-only memory* — ROM). Komputer potrafi odczytać zawartość takiej pamięci, ale nie może jej w żaden sposób zmodyfikować lub zapisać w niej nowych danych. ROM jest więc pamięcią **nieulotną**, co oznacza, że zapisane na niej dane nie zostaną usunięte nawet po wyłączeniu komputera. W pamięci ROM umieszcza się zazwyczaj programy konieczne do pracy systemu. Przykładem niech będzie program startowy, który uruchamia się zaraz po włączeniu komputera.

Nośniki danych

Nośnik danych to taki rodzaj pamięci, w której można zapisać i przechowywać dane przez bardzo długi czas — nawet wtedy, gdy komputer jest wyłączony. Na takich urządzeniach zapisane są programy, które — kiedy zajdzie taka potrzeba — są ładowane do pamięci głównej. Na nośnikach danych są także zapisywane ważne dane, takie jak dokumenty tekstowe, informacje dotyczące wynagrodzeń pracowników czy wykazy zapasów magazynowych.

Najpowszechniejszym przykładem nośnika danych jest dysk twardy. Tradycyjny **dysk twardy** zapisuje dane na nośnikach magnetycznych w postaci talerzy. Natomiast zyskujące obecnie coraz większą popularność **dyski SSD** (ang. *solid state drive*) zapisują dane w pamięciach będących układami scalonymi. W dysku SSD nie ma żadnych ruchomych elementów i działa on znacznie szybciej od tradycyjnego dysku twardego. Większość komputerów jest wyposażona w jakiś rodzaj nośnika danych — czy to tradycyjny dysk twardy, czy dysk SSD. Istnieją także zewnętrzne dyski, które można podłączyć do któregoś z portów komunikacyjnych komputera. Używa się ich najczęściej do przechowywania kopii zapasowych danych lub podczas przenoszenia danych do innego komputera.

Poza dyskami zewnętrznymi istnieje również szereg urządzeń służących do kopiowania lub przenoszenia danych pomiędzy komputerami. Są to **napędy USB** (ang. *universal serial bus drive*), mające postać małego urządzenia podłączanego do portu USB. Po ich podłączeniu system wykrywa je jako kolejny dysk — jednak w ich wnętrzu tak naprawdę nie kryją się żadne dyski. Urządzenia te zapisują dane w specjalnej pamięci zwanej **pamięcią flash**. Napędy USB (określane często jako **pendrive**) są urządzeniami niedrogimi, niezawodnymi i na tyle małymi, że mieszczą się w kieszeni spodni.

Do przechowywania danych wykorzystuje się także nośniki optyczne, takie jak dyski **CD** (ang. *compact disc*) czy **DVD** (ang. *digital versatile disc*). Dane na dyskach optycznych nie są zapisane w sposób magnetyczny, lecz zakodowane za pomocą szeregu rowków wytłoczonych na powierzchni płyty. Napędy CD i DVD odczytują zakodowane dane za pomocą lasera. Na dysku optycznym można zapisać ogromne ilości danych, a ponieważ nie ma obecnie problemu z dostępnością zapisywalnych płyt CD lub DVD, stanowią one idealny nośnik do przechowywania kopii zapasowych danych.

UWAGA: Obecnie popularność zyskuje przechowywanie danych w **chmurze**. Zapisane w chmurze dane są przechowywane na zdalnych komputerach dostępnych przez internet lub w prywatnej sieci danej firmy. Po zapisaniu danych w chmurze możesz uzyskać do nich dostęp z wielu różnych urządzeń oraz z dowolnego miejsca, pod warunkiem że masz dostęp do sieci. W chmurze można również zapisywać kopie zapasowe ważnych danych znajdujących się na komputerze.

Urządzenia wejściowe

Dane wejściowe (ang. *input*) to dane, które komputer pobiera od użytkownika lub z innego urządzenia. Urządzenie, które pobiera te dane i przesyła je do komputera, nazywamy **urządzeniem wejściowym** (ang. *input device*). Najpopularniejsze urządzenia wejściowe to myszka, klawiatura, ekran dotykowy, skaner, mikrofon i aparat cyfrowy. Dyski twarde i napędy optyczne także można traktować jako urządzenia wejściowe, ponieważ programy i dane są z nich ładowane do pamięci komputera.

Urządzenia wyjściowe

Dane wyjściowe (ang. *output*) to dane, które komputer prezentuje użytkownikowi lub przekazuje do innego urządzenia. Może to być raport sprzedaży, lista nazwisk lub plik graficzny. Dane przekazywane są do **urządzenia wyjściowego** (ang. *output device*), które z kolei je formatuje i prezentuje użytkownikowi. Popularnymi przykładami urządzeń wyjściowych są monitor lub drukarka. Dyski twarde i nagrywarki CD także można traktować jako urządzenia wyjściowe, ponieważ komputer wysyła do nich dane, które mają zostać zapisane.

Punkt kontrolny

1.1. Co to jest program komputerowy?

1.2. Co to jest sprzęt?

1.3. Wymień pięć głównych elementów komputera.

1.4. Który z komponentów komputera służy do wykonywania programu?

1.5. Który element służy jako przestrzeń robocza komputera, w której na czas działania programu zapisywane są dane i sam program?

1.6. Który element komputera służy do przechowywania danych przez dłuższy okres czasu, nawet wtedy, gdy komputer jest wyłączony?

1.7. Który element komputera pobiera dane od użytkownika lub z innego urządzenia?

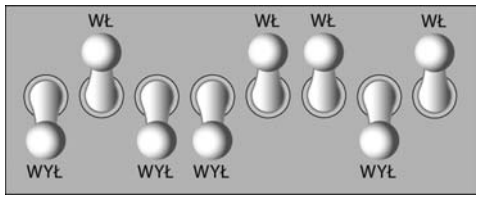
1.8. Który element komputera odpowiedzialny jest za formatowanie danych i prezentowanie ich użytkownikowi?

1.3 W jaki sposób komputer przechowuje dane

WYJAŚNIENIE: Wszelkie dane, które mają zostać zapisane w komputerze, są konwertowane do postaci sekwencji złożonej z zer (0) i jedynek (1).

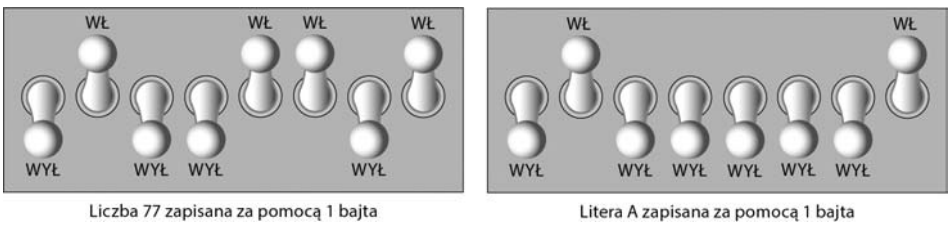
Pamięć komputera jest podzielona na malutkie obszary zwane **bajtami**. Jeden bajt pozwala zapisać jedną literę lub niewielką liczbę. Aby komputer mógł zapisać jakąś bardziej znaczącą informację, musi on być wyposażony w bardzo dużą liczbę takich bajtów. Obecnie większość komputerów jest wyposażona w pamięci o rozmiarze wielu milionów, a nawet miliardów bajtów.

Każdy bajt jest z kolei podzielony na osiem mniejszych części zwanych **bitami**. Słowo „bit" **oznacza cyfrę binarną** (ang. *binary digit*). Często stosuje się analogię, w której bit przedstawiony jest jako przełącznik — może on być włączony albo wyłączony. Jednak same bity nie są przełącznikami — przynajmniej nie w dosłownym tego słowa znaczeniu. W większości systemów komputerowych bit przyjmuje formę elementu elektronicznego, który charakteryzuje się dodatnim lub ujemnym ładunkiem elektrycznym. Dodani ładunek można sobie wyobrazić jako przełącznik w pozycji **włączonej**, a ładunek ujemny jako przełącznik w pozycji **wyłączonej**. Na rysunku 1.6 przedstawiłem coś, co można sobie wyobrazić jako 1 bajt pamięci: szereg przełączników, z których każdy jest włączony albo wyłączony.



Rysunek 1.6. Wyobraź sobie bajt jako zespół ośmiu przełączników

Kiedy komputer zapisuje w danym bajcie jakąś informację, ustawia on w odpowiedni sposób każdy z tych ośmiu przełączników. Przykładowo na rysunku 1.7 z lewej strony przedstawiłem, jak zostanie zapisana w bajcie liczba 77, natomiast z prawej strony pokazałem, jak zapisana zostanie litera A. Za chwilę dowiesz się, jak można określić, jaką wartość reprezentuje dana kombinacja ustawień przełączników.



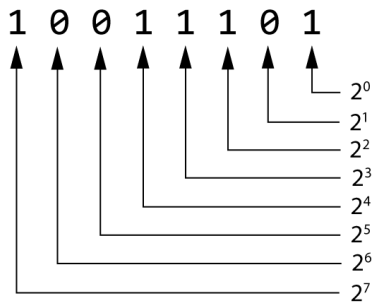
Rysunek 1.7. Kombinacje przełączników przedstawiające liczbę 77 i literę A

Zapisywanie liczb

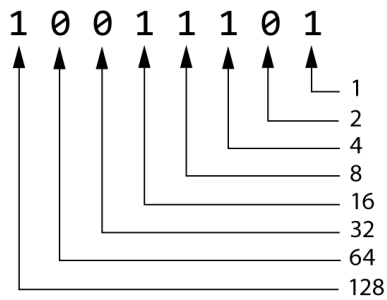
Pojedynczy bit może reprezentować liczbę w bardzo ograniczony sposób. W zależności od tego, czy bit jest ustawiony czy nie, reprezentuje on jedną z dwóch wartości. W przypadku systemów komputerowych bit, który nie jest ustawiony, reprezentuje liczbę 0, a bit ustawiony reprezentuje liczbę 1. Doskonale wpisuje się to w naturę **binarnego systemu liczbowego**. W systemie takim każdą liczbę przedstawia się za pomocą sekwencji zer i jedynek. Oto przykład liczby zapisanej w systemie binarnym:

```
10011101
```

Położenie każdej z cyfr wskazuje jednocześnie na wartość, jaką ona reprezentuje. Zaczynając od cyfry położonej najdalej z prawej strony i idąc w lewą stronę, są to kolejno wartości: 2^0, 2^1, 2^2, 2^3 itd. — przedstawiłem to na rysunku 1.8. Na rysunku 1.9 widoczna jest ta sama sekwencja, tylko z wyliczonymi już kolejnymi wartościami bitów. Zaczynając od cyfry położonej najdalej z prawej strony i idąc w lewą stronę, są to kolejno wartości: 1, 2, 4, 8 itd.

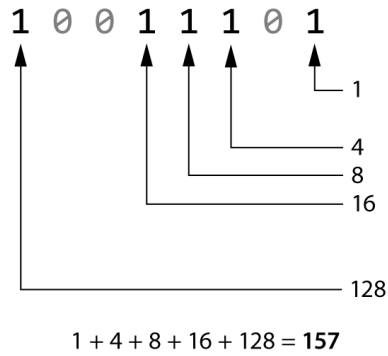


Rysunek 1.8. Wartości bitów to kolejne potęgi liczby 2



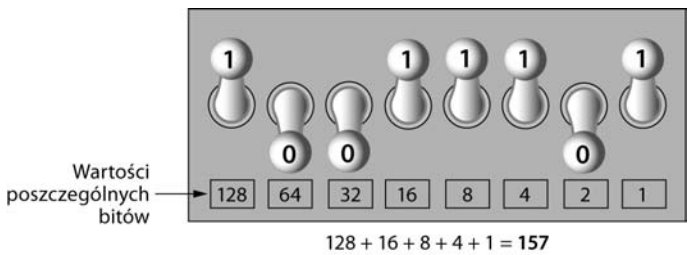
Rysunek 1.9. Wartości bitów

Aby określić, jaką wartość reprezentuje dana sekwencja bitów, należy dodać do siebie wartości wszystkich bitów ustawionych na 1. Przykładowo w przypadku liczby binarnej 10011101 jedynki znajdują się na pozycjach reprezentujących wartości 1, 4, 8, 16 i 128. Zilustrowałem to na rysunku 1.10. Suma poszczególnych wartości da nam liczbę 157. Tak więc wartość binarnej liczby 10011101 równa jest 157.



Rysunek 1.10. Określanie wartości liczby binarnej 10011101

Na rysunku 1.11 pokazałem, w jaki sposób możesz sobie wyobrazić liczbę 157 zapisaną w 1 bajcie pamięci. Każdą jedynkę reprezentuje bit w pozycji włączonej, a każe zero — bit w pozycji wyłączonej.

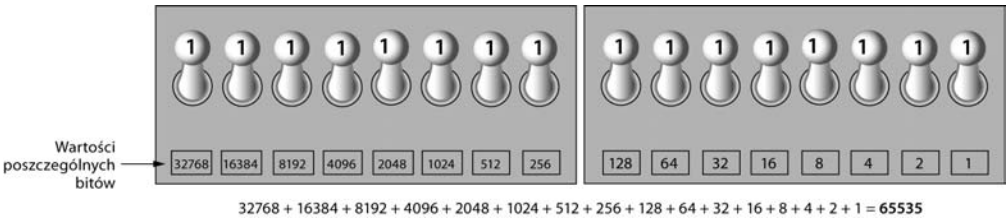


Rysunek 1.11. Kombinacja bitów reprezentująca liczbę 157

Kiedy wszystkie bity ustawione są na 0 (wyłączone), wartość bajta jest równa 0. Kiedy wszystkie bity ustawione są na 1 (włączone), wartość bajta jest równa największej wartości, jaką można w nim zapisać. Największa wartość, jaką można zapisać w 1 bajcie to 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255. Ograniczenie to wynika z faktu, że bajt składa się tylko z 8 bitów.

A co w sytuacji, gdy chcesz zapisać liczbę większą niż 255? Odpowiedź na to pytanie jest prosta: wykorzystaj więcej niż 1 bajt. Przykładowo wykorzystajmy 2 bajty — da nam to w sumie 16 bitów. W tym przypadku wartości kolejnych bitów będą równe: 2^0, 2^1, 2^2, 2^3 itd. — aż do 2^{15}. Na rysunku 1.12 pokazałem, że największa wartość, jaką można zapisać za pomocą 2 bajtów, wynosi 65535. Jeżeli będziemy chcieli zapisać jeszcze większą wartość, będziemy musieli użyć jeszcze większej liczby bajtów.

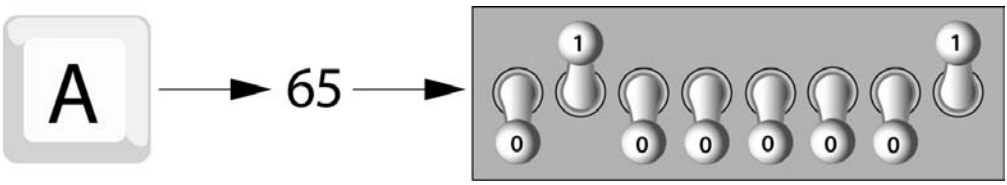
WSKAZÓWKA: Jeśli przytłaczają Cię te wszystkie informacje, nie przejmuj się! Pisząc programy, nie musimy zamieniać liczb na wartości w systemie binarnym. Jednak wiedza dotycząca tego, w jaki sposób komputer zapisuje informacje, ułatwi Ci naukę i sprawi, że będziesz lepszym programistą.



Rysunek 1.12. Większą liczbę można zapisać za pomocą 2 bajtów

Zapisywanie znaków

Każda informacja w pamięci komputera musi zostać zapisana za pomocą liczby binarnej. Dotyczy to także znaków — takich jak litery czy znaki przestankowe. Kiedy komputer zapisuje w pamięci znak, zamienia go najpierw do postaci kodu numerycznego. Następnie taki kod numeryczny zapisywany jest w pamięci komputera za pomocą liczby binarnej. Na przestrzeni lat rozwinęło się wiele różnych sposobów kodowania znaków. Z historycznego punktu widzenia najważniejszym z nich jest system **ASCII**, który jest akronimem nazwy *American Standard Code for Information Interchange*. Kodowanie ASCII to zestaw 128 kodów, które reprezentują litery w języku angielskim, znaki przestankowe oraz kilka innych znaków. Przykładowo dużą literę A reprezentuje kod o numerze 65. Jeżeli więc wpiszesz na klawiaturze literę A, w pamięci komputera zostanie zapisana liczba 65 (oczywiście za pomocą systemu binarnego). Przedstawiłem to na rysunku 1.13.



Rysunek 1.13. Litera A jest zapisana w pamięci jako liczba 65

WSKAZÓWKA: Skrót ASCII wymawia się „aski”.

Jeśli Cię to interesuje, to literę B reprezentuje liczba 66, literę C — liczba 67 itd. W dodatku A znajduje się tabela kodów ASCII oraz wartości, jakim one odpowiadają.

Zestaw znaków ASCII został stworzony na początku lat sześćdziesiątych XX wieku i ostatecznie zyskał aprobatę większości producentów sprzętu komputerowego. Jest to jednak zestaw bardzo ograniczony — można za jego pomocą zakodować tylko 128 znaków. Aby temu zaradzić, na początku lat dziewięćdziesiątych XX wieku stworzono nowy zestaw znaków, o nazwie **Unicode**. Jest to bardzo obszerny zestaw, który jednocześnie jest kompatybilny z systemem ASCII, ale można za jego pomocą kodować znaki z bardzo wielu języków. Obecnie Unicode jest standardowym system kodowania znaków w przemyśle komputerowym.

Zapisywanie bardziej skomplikowanych liczb

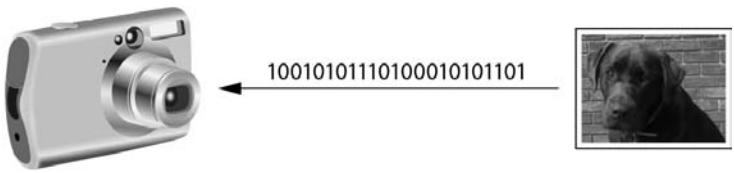
Przed chwilą wyjaśniłem Ci, jak w pamięci przechowywane są wartości liczbowe. Być może zwróciłeś wtedy uwagę, że za pomocą systemu binarnego można wyrażać tylko liczby całkowite — począwszy od 0. Za pomocą opisanej przeze mnie prostej techniki nie da się przedstawić liczb ujemnych i zmiennoprzecinkowych (np. 3,14159).

Komputery jednak potrafią zapisywać w pamięci zarówno liczby ujemne, jak i zmiennoprzecinkowe, ale aby to było możliwe, poza system binarnym korzysta się także z różnych metod kodowania. Na przykład liczby ujemne koduje się za pomocą techniki zwanej **uzupełnieniem do 2**, a ułamki — za pomocą **zapisu zmiennoprzecinkowego** (ang. *floating-point notation*). Nie musisz wiedzieć, jak działają te kodowania — wystarczy Ci wiedza, że służą one do zapisywania w systemie binarnym liczb ujemnych i ułamkowych.

Inne typy danych

Komputery są często określane mianem urządzeń cyfrowych. Przymiotnika „cyfrowy" używamy w przypadku rzeczy, które w jakikolwiek sposób korzystają z liczb binarnych. **Cyfrowe dane** (ang. *digital data*) to dane zapisane w systemie binarnym, a **urządzenie cyfrowe** to urządzenie, które wykorzystuje podczas działania dane binarne. W tym podrozdziale omówiłem, w jaki sposób w systemie binarnym zapisywane są liczby i znaki, jednak komputery operują także na wielu innych typach danych cyfrowych.

Wyobraź sobie zdjęcia, jakie wykonujesz swoim aparatem cyfrowym. Obrazy te składają się z malutkich kolorowych kropek zwanych **pikselami** (słowo to pochodzi od angielskiego wyrażenia *picture element*). Na rysunku 1.14 pokazałem, że każdy piksel obrazu zamieniany jest na liczbę, która odpowiada jego kolorowi. Liczba ta jest zapisywana w pamięci za pomocą systemu binarnego.



Rysunek 1.14. Obraz cyfrowy jest zapisywany w systemie binarnym (zdjęcie dzięki uprzejmości Gaddis, Tony)

Muzyka, którą odtwarzasz z płyt CD, iPoda czy odtwarzacza MP3, też ma postać cyfrową. Utwór zapisany cyfrowo jest podzielony na malutkie kawałki zwane **próbkami**. Każda próbka jest zamieniana na wartość binarną, która z kolei może zostać zapisana w pamięci. Z im większej liczby próbek składa się dany utwór muzyczny, tym dokładniej odzwierciedla on oryginalne nagranie. Jedna sekunda utworu o jakości CD składa się z ponad 44000 próbek!

Punkt kontrolny

1.9. Ile pamięci będziesz potrzebować, aby zapisać niewielką liczbę lub jedną literę alfabetu?

1.10. Jak inaczej określamy mały „przełącznik", który można ustawić w pozycji włączonej bądź wyłączonej?

1.11. Jak nazywa się system liczbowy, w którym wartości zapisujemy za pomocą szeregu zer i jedynek?

1.12. Do czego służy ASCII?

1.13. Za pomocą którego systemu kodowania można przedstawić wszystkie znaki wszystkich języków świata?

1.14. Co oznaczają wyrażenia „dane cyfrowe" i „urządzenie cyfrowe"?

1.4 W jaki sposób działa program

WYJAŚNIENIE: Procesor komputera potrafi zrozumieć jedynie instrukcje zapisane w języku maszynowym. Ponieważ język ten jest na tyle trudny, że człowiekowi bardzo ciężko byłoby napisać za jego pomocą cały program, stworzono inne języki programowania komputerów.

Wcześniej wspomniałem, że procesor jest najważniejszym elementem komputera, ponieważ to on odpowiada za wykonanie programu. Czasami mówi się, że procesor to „mózg komputera", jednak jest to tylko popularna metafora, a sam procesor jest po prostu urządzeniem elektronicznym, którego zadaniem jest wykonywanie określonych operacji — w szczególności takich jak:

- odczytywanie danych z pamięci głównej;
- dodawanie do siebie dwóch liczb;
- odejmowanie jednej liczby od drugiej;
- mnożenie dwóch liczb;
- dzielenie jednej liczby przez drugą;
- przenoszenie danych z jednego obszaru pamięci do innego;
- sprawdzanie, czy dana wartość jest równa innej wartości;
- itp.

Zapewne wywnioskowałeś z tej listy, że procesor zajmuje się głównie wykonywaniem na danych prostych operacji. Jednak sam z siebie nie jest on w stanie wykonać żadnej operacji — trzeba mu powiedzieć, co ma rozbić, i właśnie to jest zadaniem programu. Program jest niczym innym jak zbiorem instrukcji, które ma wykonać procesor.

Każda instrukcja w programie to polecenie wykonania określonej operacji wydane procesorowi. Oto przykład instrukcji, jaka może wystąpić w programie:

```
10110000
```

Zarówno dla mnie, jak i dla Ciebie jest to szereg zer i jedynek. Jednak dla procesora jest to polecenie wykonania określonej operacji[1]. Ponieważ procesor rozumie jedynie instrukcje zapisane w języku maszynowym, jest ona zapisana za pomocą zer i jedynek — polecenia w **języku maszynowym** (ang. *machine language*) są zawsze zapisane w systemie binarnym.

Dla każdej operacji, którą potrafi wykonać dany procesor, istnieje osobna instrukcja w języku maszynowym. Przykładowo istnieje osobna instrukcja dla dodawania liczb, osobna dla odejmowania liczb itd. Kompletny zestaw instrukcji, które potrafi wykonać dany procesor, nazywa się **listą rozkazów procesora** (ang. *instruction set*).

UWAGA: Obecnie istnieje wielu producentów procesorów komputerowych — najbardziej znanymi są firmy Intel, AMD i Motorola. Jeżeli spojrzysz na swój komputer, być może znajdziesz na nim naklejkę z logo danego producenta.

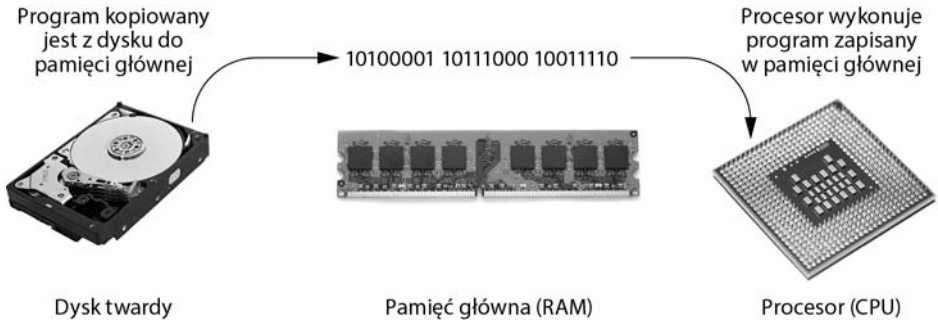
Każda rodzina procesorów charakteryzuje się inną listą rozkazów procesora, którą potrafią zrozumieć jedynie procesory tej rodziny. Przykładowo procesory Intel rozumieją instrukcje pochodzące z ich listy rozkazów, ale nie będą w stanie zrozumieć instrukcji przeznaczonych dla procesorów Motorola.

Instrukcja w języku maszynowym, którą przedstawiłem wcześniej, to tylko jeden z przykładów. Aby komputer mógł wykonać jakąś sensowną czynność, potrzebnych będzie znacznie więcej instrukcji. Ponieważ sam procesor rozumie jedynie bardzo elementarne rozkazy, będzie on musiał wykonać ich bardzo wiele, aby wykonać jakieś znaczące działanie. Przykładowo, jeśli chcemy, aby komputer obliczył wartość odsetek, jakie zostaną naliczone w ciągu roku na rachunku oszczędnościowym, procesor będzie musiał wykonać bardzo dużą liczbę instrukcji — ponadto instrukcje te będą musiały być uruchomione w odpowiedniej kolejności. Nie jest niczym niezwykłym program składający się z tysięcy, a nawet wielu milionów instrukcji w języku maszynowym.

Programy są zazwyczaj zapisane na nośnikach danych — na przykład na dysku twardym. Kiedy instalujesz na komputerze program, jest on kopiowany z płyty CD lub z pliku pobranego z sieci na dysk twardy komputera.

Pomimo faktu, że dany program jest już zapisany na dysku twardym, za każdym razem, gdy procesor ma go wykonać, program ten musi najpierw zostać skopiowany do pamięci głównej komputera. Powiedzmy, że na dysku twardym znajduje się edytor tekstowy. Aby go uruchomić, musisz kliknąć myszką odpowiednią ikonkę. Spowoduje to skopiowanie programu z dysku twardego do pamięci głównej. Następnie procesor przystąpi do wykonywania kopii programu zapisanej już w pamięci głównej komputera. Proces ten przedstawiłem na rysunku 1.15.

[1] Przedstawiony przykład to prawdziwa instrukcja dla procesora Intel. Nakazuje ona przeniesienie wartości do procesora.

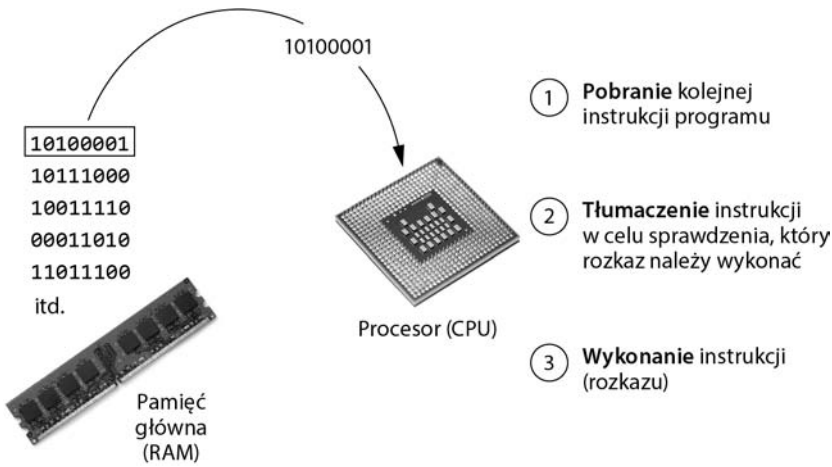


Rysunek 1.15. Program najpierw jest kopiowany do pamięci głównej, a następnie uruchamiany (dzięki uprzejmości Lefteris Papaulakis/Shutterstock, Garsya/Shutterstock oraz marpan/Shutterstock)

Kiedy procesor przechodzi do wykonywania kolejnych instrukcji zapisanych w programie, dochodzi do tak zwanego **cyklu rozkazowego**. Cykl ten składa się z trzech faz i jest powtarzany dla każdej instrukcji w programie. Fazy te są następujące:

1. **Pobranie**. Program składa się z długiej sekwencji instrukcji w języku maszynowym. Pierwsza faza cyklu rozkazowego ma za zadanie pobrać (odczytać) kolejną instrukcję z pamięci głównej i przekazać ją do procesora.
2. **Tłumaczenie**. Instrukcja zapisana w języku maszynowym to liczba w systemie binarnym reprezentująca określony rozkaz, który ma wykonać procesor. W tej fazie procesor tłumaczy instrukcję pobraną z pamięci i sprawdza, który rozkaz powinien wykonać.
3. **Wykonanie**. W ostatniej fazie cyklu procesor wykonuje dany rozkaz.

Rysunek 1.16 ilustruje te trzy fazy.



Rysunek 1.16. Cykl rozkazowy (dzięki uprzejmości Garsya/Shutterstock i marpan/Shutterstock)

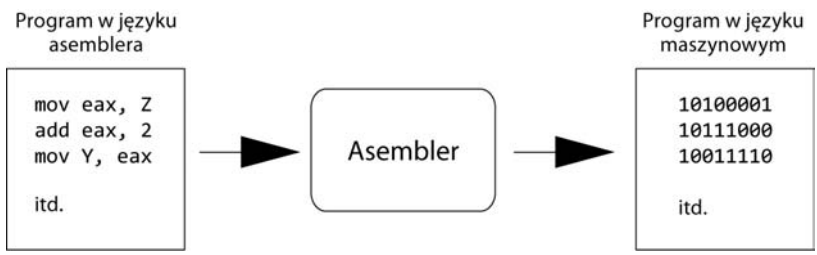
Z języka maszynowego na język asemblera

Komputer potrafi wykonywać tylko programy zapisane w języku maszynowym. Jak wspomniałem wcześniej, taki program może składać się z tysięcy, a nawet milionów instrukcji zapisanych w systemie binarnym i jego pisanie byłoby zajęciem bardzo nudnym i czasochłonnym. Byłoby to także bardzo trudne, gdyż postawienie zera lub jedynki w nieodpowiednim miejscu spowodowałoby błąd programu.

Procesor komputera rozumie jedynie instrukcje zapisane w języku maszynowym, jednak dla człowieka pisanie programu w tym języku jest niepraktyczne. Z tego powodu u zarania dziejów komputerów[2] wymyślono język zwany **językiem asemblera**. Zamiast z liczb w systemie binarnym korzysta się w nim z kilkuliterowych skrótów zwanych **mnemonikami**. Przykładowo w języku asemblera mnemonika `add` oznacza operację dodawania, mnemonika `mul` oznacza mnożenie, a mnemonika `mov` oznacza operację przeniesienia wartości do określonego miejsca w pamięci. Kiedy programista pisał program w języku asemblera, posługiwał się takimi skrótowymi mnemonikami zamiast liczbami w systemie binarnym.

> **UWAGA:** Istnieje wiele różnych odmian języka asemblera. Wspomniałem wcześniej, że każda rodzina procesorów charakteryzuje się swoją własną listą rozkazów. Analogicznie zazwyczaj każda rodzina procesorów ma swój własny język asemblera.

Procesor komputerowy nie jest jednak w stanie wykonać programu zapisanego w języku asemblera — rozumie on jedynie język maszynowy. Z tego powodu stworzono specjalne programy zwane **asemblerami**, których zadaniem jest przetłumaczenie programu napisanego w języku asemblera na program w języku maszynowym. Proces ten przedstawiłem na rysunku 1.17. Program w języku maszynowym wygenerowany przez asembler może już zostać wykonany przez procesor.



Rysunek 1.17. Asembler tłumaczy program zapisany w języku asemblera na program w języku maszynowym

[2] Pierwszy język asemblera został stworzony najprawdopodobniej w latach czterdziestych XX wieku na Uniwersytecie Cambridge i korzystano z niego podczas pracy z komputerem EDSAC.

Języki wysokiego poziomu

Pomimo faktu, że dzięki językowi asemblera nie ma już konieczności pisania programu w języku maszynowym, nie jest on pozbawiony pewnych wad. Język asemblera pełni głównie rolę zastępczą dla języka maszynowego i podobnie jak w jego przypadku wymaga od programisty dobrej znajomości danego procesora. Ponadto, aby napisać nawet najprostszy program w języku asemblera, trzeba użyć bardzo wielu instrukcji. Ponieważ z natury język asemblera jest bardzo bliski językowi maszynowemu, nazywa się go **językiem niskiego poziomu** (ang. *low-level language*).

W latach pięćdziesiątych XX wieku pojawiła się nowa generacja języków programowania zwanych **językami wysokiego poziomu** (ang. *high-level language*). To właśnie dzięki nim możemy tworzyć potężne i złożone programy, bez konieczności posiadania wiedzy na temat samego procesora, a jednocześnie nie potrzebujemy do tego ogromnej liczby niskopoziomowych instrukcji. Ponadto większość języków wysokiego poziomu zawiera łatwe do zrozumienia słowa. Przykładowo, kiedy programista piszący program w języku COBOL (jeden z pierwszych języków wysokiego poziomu, stworzony w latach pięćdziesiątych XX wieku) chciał wyświetlić na ekranie komputera komunikat „Hello world", wystarczyło, że użył takiej oto instrukcji:

```
Display "Hello world"
```

Ta sama operacja w języku asemblera wymagałaby użycia wielu instrukcji i wiedzy na temat sposobu komunikowania się procesora z monitorem. Ten przykład pokazuje, że dzięki językom wysokiego poziomu programista może się skoncentrować na tym, jakie zadania powinien wykonać program, a nie na tym, jak procesor będzie wykonywał dany program.

Od tamtego czasu powstały tysiące języków wysokiego poziomu. W tabeli 1.1 zamieściłem listę kilku najbardziej popularnych. Jeśli uczysz się na kierunku związanym z branżą informatyczną, najprawdopodobniej poznasz jeden lub kilka spośród tych języków.

Każdy język wysokiego poziomu charakteryzuje się zestawem słów, które programista musi poznać, aby mógł z niego korzystać. Słowa występujące w danym języku wysokiego poziomu nazywa się **słowami kluczowymi** (ang. *key words*) lub **słowami zarezerwowanymi** (ang. *reserved words*). Każde słowo kluczowe ma określone znaczenie i nie może zostać wykorzystane do żadnego innego celu. Wcześniej pokazałem przykład instrukcji w języku COBOL, w której występuje słowo kluczowe `Display`, służące do wyświetlania komunikatu na ekranie komputera. W języku Python do tego samego celu służy słowo kluczowe `print`.

Poza słowami kluczowymi w językach programowania występują także **operatory**, które wykonują na danych określone działania. Przykładowo w każdym języku występują operatory matematyczne, które służą do wykonywania działań arytmetycznych. W Javie i w wielu innych językach znak `+` jest operatorem służącym do dodawania do siebie dwóch liczb. Następujące wyrażenie dodaje do siebie liczby 12 i 75:

```
12 + 75
```

Tabela 1.1. Języki programowania

Język	Opis
Ada	Język Ada został stworzony w latach 70. XX w. i używany był głównie przez departament obrony Stanów Zjednoczonych. Język nazwano na cześć hrabiny Ady Lovelace, uważanej za osobę mającą ogromny wpływ na dziedzinę programowania.
BASIC	BASIC jest językiem ogólnego zastosowania, a jego nazwa to akronim od *Beginners All-purpose Symbolic Instruction Code*. Powstał w latach 60. XX w. jako prosty język do nauki programowania. Obecnie można znaleźć wiele odmian języka BASIC.
FORTRAN	Język FORmula TRANslator był pierwszym językiem programowania wysokiego poziomu. Powstał w latach 50. XX w. i służył do przeprowadzania skomplikowanych obliczeń matematycznych.
COBOL	Język COBOL został stworzony w latach 50. XX w., a jego nazwa jest akronimem od *Common Business-Oriented Language*. Służył głównie do tworzenia aplikacji biznesowych.
Pascal	Pascal powstał w 1970 r. i pierwotnie służył jako język dydaktyczny do nauki programowania. Język nazwano na cześć słynnego matematyka, fizyka i filozofa, Blaise'a Pascala.
C i C++	C i C++ (ta druga nazwa wymawiana jako „c plus plus") to języki ogólnego zastosowania o ogromnych możliwościach stworzone przez firmę Bell Labolatories. Język C powstał w 1972 r., a język C++ w 1983 r.
C#	Nazwę C# wymawia się jako „c szarp". Język ten służy do tworzenia aplikacji dla platformy .NET i powstał około 2000 r. z inicjatywy firmy Microsoft.
Java	Język Java został stworzony we wczesnych latach 90. XX w. przez firmę Sun Microsystems (której właścicielem jest obecnie Oracle). Można dzięki niemu tworzyć zarówno aplikacje działające na komputerze, jak i aplikacje internetowe działające na serwerach WWW.
JavaScript	Język JavaScript powstał w latach 90. XX w. i wykorzystywany jest do przetwarzania stron WWW. Mimo częściowego pokrewieństwa nazwy ma on niewiele wspólnego z językiem Java.
Python	Python to język ogólnego zastosowania stworzony w latach 90. XX w. Stał się popularnym narzędziem do tworzenia aplikacji biznesowych i naukowych.
Ruby	Ruby to język ogólnego zastosowania stworzony w latach 90. XX w. Zyskuje popularność jako język do tworzenia aplikacji działających na serwerach WWW.
Visual Basic	Visual Basic (często nazywany także VB) to język i środowisko programistyczne stworzone przez firmę Microsoft. Dzięki niemu można bardzo szybko tworzyć aplikacje dla systemu Windows. Język VB powstał w latach 90. XX w.

Poza słowami kluczowymi i operatorami każdy język charakteryzuje się także określoną **składnią** (ang. *syntax*), czyli zestawem zasad, których należy bezwzględnie przestrzegać, pisząc program w danym języku. Składnia wskazuje, w jaki sposób należy

używać w programie słów kluczowych, operatorów i znaków przestankowych. Podczas nauki danego języka programowania musisz poznać dokładnie jego składnię.

Poszczególne instrukcje, których będziesz używać w programie pisanym w języku wysokiego poziomu, nazywamy **poleceniami** (ang. *statements*). Polecenie może się składać ze słów kluczowych, operatorów, znaków przestankowych oraz innych dopuszczalnych elementów, ułożonych w odpowiedniej kolejności tak, aby wykonać określoną operację.

UWAGA: Języki ludzkie także charakteryzują się składnią. Pamiętasz, jak podczas nauki języka polskiego poznawałeś zasady związane z podmiotami i orzeczeniami oraz innymi częściami zdania? Uczyłeś się wtedy właśnie składni języka polskiego.

Pomimo że ludzie podczas mówienia i pisania bardzo często łamią zasady składni, ich rozmówcy nie mają raczej problemu ze zrozumieniem. Niestety komputery są pozbawione tej cechy. Jeżeli pisząc program złamiesz chociaż jedną zasadę składni, program się nie uruchomi.

Kompilatory i interpretery

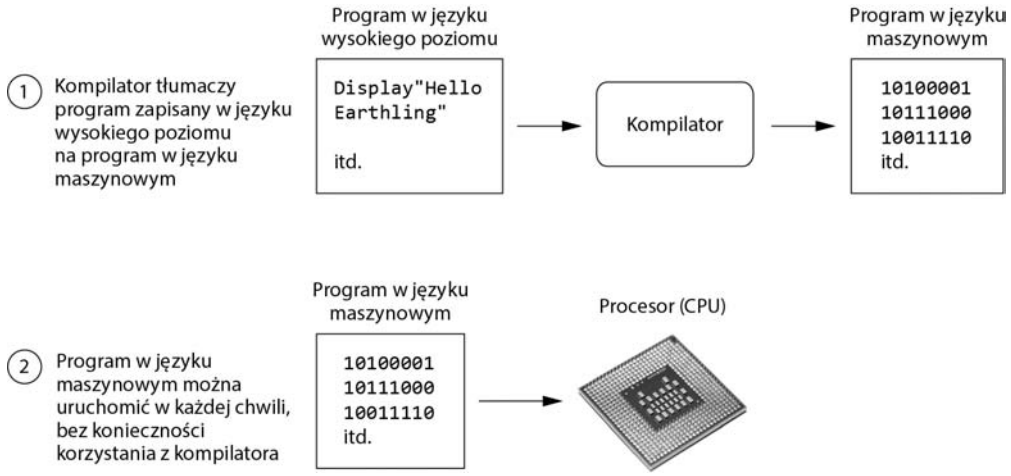
Ponieważ procesor komputera rozumie jedynie rozkazy w języku maszynowym, programy napisane w językach wysokiego poziomu muszą zostać najpierw przetłumaczone na język maszynowy. Po napisaniu programu w języku wysokiego poziomu programista musi skorzystać z kompilatora lub interpretera, aby przetłumaczyć program na język maszynowy.

Kompilator to program, którego zadaniem jest przetłumaczenie programu napisanego w języku wysokiego poziomu na program w języku maszynowym. Taki program w języku maszynowym może być następnie uruchomiony w dowolnej chwili. Przedstawiłem to na rysunku 1.18. Na rysunku widać, że kompilacja i uruchomienie programu to dwa odrębne procesy.

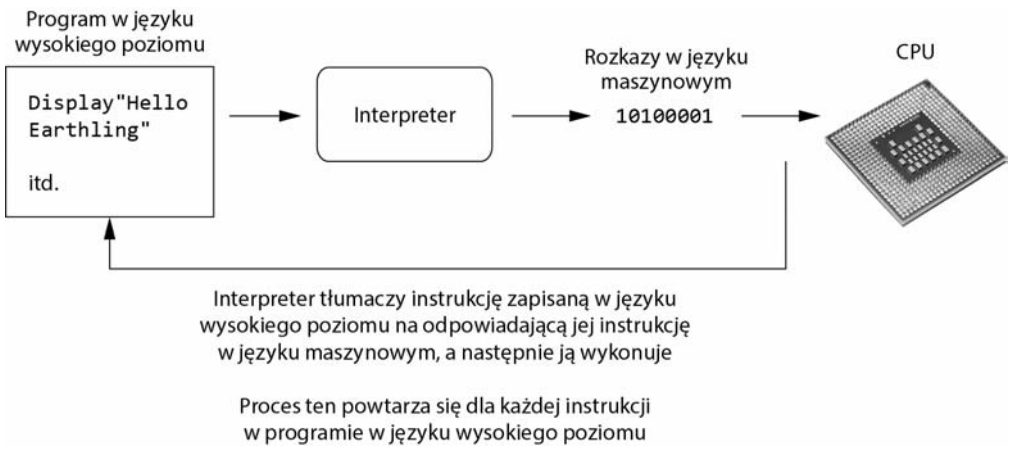
Interpreter to program, który jednocześnie tłumaczy i wykonuje instrukcje w języku wysokiego poziomu. Interpreter odczytuje pojedyncze polecenie w programie, następnie tłumaczy je na język maszynowy i natychmiast wykonuje wynikowy kod maszynowy. Działanie to odbywa się dla każdego polecenia w programie. Proces ten pokazałem na rysunku 1.19. Ponieważ interpretery łączą tłumaczenie kodu i jego wykonanie, nie tworzą zazwyczaj osobnego programu w języku maszynowym.

UWAGA: Ponieważ skompilowany program w momencie uruchomienia jest już w całości przetłumaczony na język maszynowy, działa on szybciej od programu uruchomionego za pomocą interpretera.

Polecenia w języku wysokiego poziomu, które tworzy programista, nazywamy **kodem źródłowym** lub po prostu **kodem**. Zazwyczaj programista pisze program w pliku tekstowym, po czym zapisuje go na dysku twardym. Następnie musi za pomocą kompilatora



Rysunek 1.18. Kompilowanie programu zapisanego w języku wysokiego poziomu i jego uruchomienie (dzięki uprzejmości marpan/Shutterstock)



Rysunek 1.19. Wykonywanie programu w języku wysokiego poziomu za pomocą interpretera (dzięki uprzejmości marpan/Shutterstock)

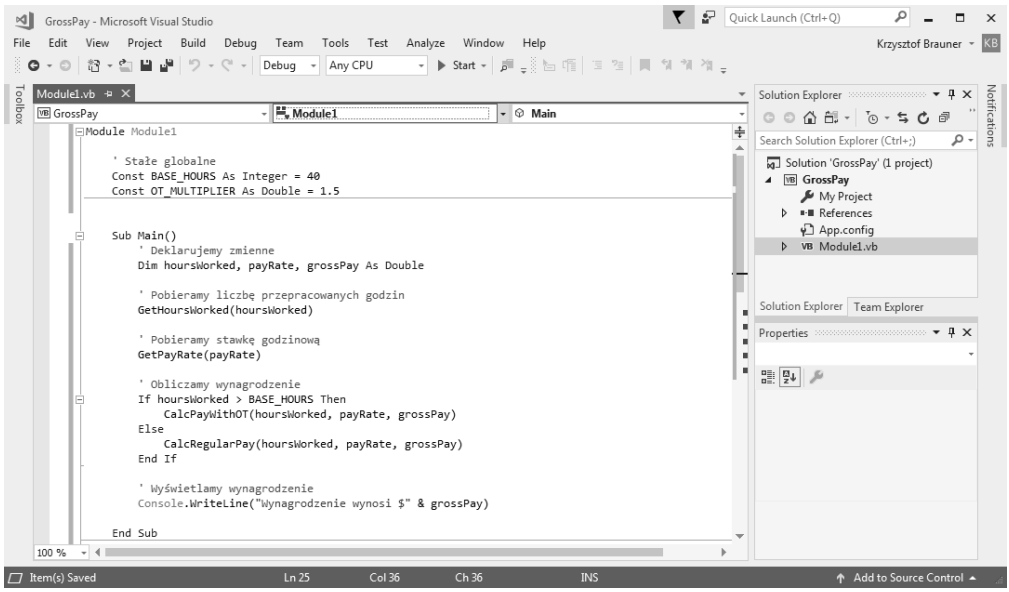
przetłumaczyć kod na program w języku maszynowym lub skorzystać z interpretera, aby przetłumaczyć i wykonać kod programu. Jeśli jednak w programie znajdą się jakieś błędy składniowe, nie będzie on mógł zostać przetłumaczony. **Błąd składniowy** (ang. *syntax error*) to błąd wynikający najczęściej z literówki, brakującego znaku przestankowego lub nieprawidłowego zastosowania operatora. W momencie wystąpienia takiego błędu kompilator lub interpreter wyświetli komunikat informujący o błędzie składniowym w kodzie programu. Programista musi wtedy poprawić błąd i ponownie przystąpić to tłumaczenia programu na język maszynowy.

Zintegrowane środowisko programistyczne (IDE)

Chociaż do tworzenia programu można wykorzystać edytor tekstowy taki jak Notatnik (będący częścią systemu operacyjnego Windows), większość programistów korzysta ze specjalnego oprogramowania, zwanego **zintegrowanym środowiskiem programistycznym** (ang. *integrated development environment* — IDE). Większość takich środowisk składa się z następujących elementów:

- edytor tekstowy wyposażony w specjalne narzędzia ułatwiające pisanie poleceń w danym języku wysokiego poziomu;
- kompilator lub interpreter;
- narzędzia do testowania programów i wychwytywania błędów.

Na rysunku 1.20 przestawiłem ekran programu Microsoft Visual Studio — popularnego środowiska służącego do tworzenia programów w językach C++, Visual Basic i C#. Innymi przykładami zintegrowanych środowisk programistycznych są pakiety Eclipse, NetBeans, Dev-C++ i jGRASP.



Rysunek 1.20. Zintegrowane środowisko programistyczne (dzięki uprzejmości Microsoft Corporation)

Punkt kontrolny

1.15. W którym języku muszą być zapisane instrukcje, aby zrozumiał je procesor?

1.16. Do której pamięci kopiowany jest program, zanim procesor zacznie go wykonywać?

1.17. Jaki proces następuje, kiedy procesor przechodzi do wykonywania kolejnych instrukcji zapisanych w programie?

1.18. Co to jest język asemblera?

1.19. Który rodzaj języków programowania umożliwia tworzenie potężnych i złożonych programów, bez konieczności posiadania wiedzy na temat procesora?

1.20. Każdy język programowania charakteryzuje się zestawem zasad, których należy przestrzegać, pisząc program w danym języku. Jak nazywa się ten zestaw zasad?

1.21. Jak nazywa się program, którego zadaniem jest przetłumaczenie programu zapisanego w języku wysokiego poziomu na program w języku maszynowym?

1.22. Jak nazywa się program, który jednocześnie tłumaczy i wykonuje kolejne instrukcje programu zapisanego w języku wysokiego poziomu?

1.23. Jak nazywa się błąd wynikający z literówki w słowie kluczowym, brakującego znaku przestankowego czy niewłaściwego użycia operatora?

1.5 Rodzaje oprogramowania

WYJAŚNIENIE: Programy zazwyczaj zaliczają się do jednej z dwóch kategorii: oprogramowania systemowego lub oprogramowania użytkowego. Oprogramowanie systemowe składa się z szeregu programów, które odpowiadają za pracę komputera i powiększają jego możliwości. Oprogramowanie użytkowe sprawia, że komputer staje się przydatnym narzędziem w codziennej pracy.

Aby działać, komputer potrzebuje oprogramowania. Wszystko, co robi komputer — od momentu, gdy go włączysz, aż do chwili jego wyłączenia — odbywa się dzięki oprogramowaniu. Istnieją dwie kategorie oprogramowania: oprogramowanie systemowe i oprogramowanie użytkowe. Większość programów komputerowych można z łatwością zaklasyfikować do jednej z tych kategorii. Przyjrzyjmy się więc im bliżej.

Oprogramowanie systemowe

Programy, które odpowiadają za podstawowe operacje komputera, nazywa się **oprogramowaniem systemowym**. Składają się na nie następujące programy:

- **System operacyjny.** System operacyjny to zestaw najważniejszych programów zapewniających pracę komputera. Zadaniem systemu operacyjnego jest sterowanie sprzętem, z jakiego składa się dany komputer, zarządzanie wszystkimi urządzeniami podłączonymi do komputera, umożliwienie odczytu i zapisu danych na nośnikach danych, umożliwienie uruchomienia na komputerze innych programów. Dzisiaj najczęściej używanymi systemami operacyjnymi są Windows, Mac OS, iOS, Android i Linux.

- **Programy narzędziowe.** Programy narzędziowe wykonują specjalistyczne zadania zapewniające bezpieczeństwo danych zapisanych na komputerze lub powiększające możliwości komputera. Przykładem programów narzędziowych są programy antywirusowe, programy do kompresji danych i tworzenia kopii zapasowych.
- **Narzędzia do tworzenia oprogramowania.** Narzędzia do tworzenia oprogramowania to programy, które umożliwiają programistom tworzenie, modyfikowanie i testowanie oprogramowania. Do tej kategorii zaliczają się na przykład asemblery, kompilatory i interpretery.

Oprogramowanie użytkowe

Oprogramowaniem użytkowym nazywamy wszystkie programy, dzięki którym komputer staje się użytecznym narzędziem w codziennej pracy. To programy, przy których użytkownicy komputera spędzają najwięcej czasu. Na rysunku 1.1, znajdującym się na początku tego rozdziału, pokazałem ekrany dwóch bardzo popularnych programów użytkowych — procesora tekstu Microsoft Word oraz programu do tworzenia i wyświetlania prezentacji Microsoft PowerPoint. Innymi przykładami programów użytkowych są arkusze kalkulacyjne, programy do obsługi wiadomości e-mail, przeglądarki internetowe i gry.

Punkt kontrolny

1.24. Jak nazywa się zestaw najważniejszych programów zapewniających pracę komputera?

1.25. Jak nazywają się programy, które wykonują specjalistyczne zadania, np. programy antywirusowe, programy do kompresji danych i programy tworzenia kopii zapasowych?

1.26. Do której kategorii oprogramowania należą takie programy jak arkusze kalkulacyjne, programy do obsługi wiadomości e-mail, przeglądarki internetowe i gry?

Pytania kontrolne

Test jednokrotnego wyboru

1. _____ to szereg instrukcji, które musi uruchomić komputer, aby wykonać określone działanie.
   a) kompilator
   b) program
   c) interpreter
   d) język programowania

2. Fizyczne urządzenia, w jakie wyposażony jest komputer, nazywamy _____.
   a) sprzętem
   b) oprogramowaniem
   c) systemem operacyjnym
   d) narzędziami
3. Elementem komputera odpowiedzialnym za wykonanie programu jest _____.
   a) RAM
   b) nośnik danych
   c) pamięć główna
   d) procesor
4. Dostępne obecnie procesory mają postać małych chipów zwanych _____.
   a) ENIAC
   b) mikroprocesorami
   c) chipami pamięci
   d) systemem operacyjnym
5. W momencie uruchomienia programu zarówno program, jak i dane, na których ten program pracuje, są przechowywane w _____.
   a) nośniku danych
   b) procesorze
   c) pamięci głównej
   d) mikroprocesorze
6. Jak nazywa się pamięć ulotna, która jest wykorzystywana jako tymczasowe miejsce przechowywania danych na czas działania programu?
   a) RAM
   b) nośnik danych
   c) dysk twardy
   d) napęd USB
7. Pamięć, która potrafi przechowywać dane przez dłuższy okres czasu — nawet po wyłączeniu komputera — nazywa się _____.
   a) RAM
   b) pamięcią główną
   c) nośnikiem danych
   d) pamięcią procesora
8. Urządzenie, które pobiera dane od użytkownika lub z innego urządzenia, a następnie przekazuje je do komputera, nazywa się _____.
   a) urządzeniem wyjściowym
   b) urządzeniem wejściowym
   c) nośnikiem danych
   d) pamięcią główną
9. Monitor komputerowy jest _____.
   a) urządzeniem wyjściowym
   b) urządzeniem wejściowym
   c) nośnikiem danych
   d) pamięcią główną

10. Jeden _____ pamięci wystarczy, aby zapisać w niej literę alfabetu angielskiego lub niewielką liczbę.
    a) bajt
    b) bit
    c) przełącznik
    d) tranzystor
11. Jeden bajt składa się z ośmiu _____.
    a) procesorów
    b) instrukcji
    c) zmiennych
    d) bitów
12. W _____ systemie liczbowym liczby przedstawia się za pomocą szeregu zer i jedynek.
    a) szesnastkowym
    b) binarnym
    c) ósemkowym
    d) dziesiętnym
13. Wyłączony bit reprezentuje wartość _____.
    a) 1
    b) -1
    c) 0
    d) „nie”
14. Zestaw 128 kodów liczbowych reprezentujących litery alfabetu angielskiego, znaki przestankowe i inne znaki nazywa się _____.
    a) binarnym system liczbowym
    b) ASCII
    c) Unicode
    d) ENIAC
15. Rozbudowany zestaw kodowania znaków reprezentujący wszystkie znaki wielu języków świata nazywa się _____.
    a) binarnym systemem liczbowym
    b) ASCII
    c) Unicode
    d) ENIAC
16. Liczby ujemne zapisuje się w systemie binarnym za pomocą techniki zwanej _____.
    a) uzupełnieniem do 2
    b) zapisem zmiennoprzecinkowym
    c) ASCII
    d) Unicode
17. Ułamki zapisuje się w systemie binarnym za pomocą techniki zwanej _____.
    a) uzupełnieniem do 2
    b) zapisem zmiennoprzecinkowym
    c) ASCII
    d) Unicode

18. Malutkie kropki, z których składają się pliki graficzne, nazywa się _____.
    a) bitami
    b) bajtami
    c) zestawami kolorów
    d) pikselami
19. Jeśli przyjrzysz się programowi napisanemu w języku maszynowym, to zobaczysz _____.
    a) kod w języku Java
    b) ciąg liczb binarnych
    c) słowa w języku angielskim
    d) obwód drukowany
20. Podczas fazy _____ cyklu rozkazowego procesor sprawdza, którą operację powinien wykonać.
    a) pobierania
    b) tłumaczenia
    c) wykonania
    d) występującej tuż przed wykonaniem polecenia
21. Komputer potrafi wykonywać tylko programy napisane w języku _____.
    a) Java
    b) asemblera
    c) maszynowym
    d) C++
22. Zadaniem _____ jest przetłumaczenie programu zapisanego w języku asemblera na program w języku maszynowym.
    a) asemblera
    b) kompilatora
    c) tłumacza
    d) interpretera
23. Słowa, z których składa się język programowania wysokiego poziomu, to _____.
    a) polecenia binarne
    b) mnemoniki
    c) polecenia
    d) słowa kluczowe
24. Zestaw zasad, których należy przestrzegać, pisząc program w danym języku, nazywa się _____.
    a) składnią
    b) interpunkcją
    c) słowami kluczowymi
    d) operatorami
25. _____ to program, którego zadaniem jest przetłumaczenie programu zapisanego w języku wysokiego poziomu na program w języku maszynowym.
    a) asembler
    b) kompilator
    c) tłumacz
    d) narzędzie

Prawda czy fałsz?

1. Dostępne obecnie procesory to ogromne urządzenia mechaniczno-elektryczne składające się z lamp elektronowych i przełączników.
2. Pamięć główna to inaczej RAM.
3. Każda informacja, która ma zostać zapisana w pamięci komputera, musi mieć postać liczby w systemie binarnym.
4. Zdjęć wykonanych aparatem cyfrowym nie da się zapisać za pomocą liczb w systemie binarnym.
5. Jedynym językiem, jaki potrafi zrozumieć procesor komputera, jest język maszynowy.
6. Język asemblera jest językiem wysokiego poziomu.
7. Interpreter to program, którego zadaniem jest jednoczesne tłumaczenie i wykonywanie programu zapisanego za pomocą języka wysokiego poziomu.
8. Program, w którym występuje błąd składniowy, można skompilować i uruchomić.
9. Windows, Mac OS, iOS, Android i Linux to przykłady programów użytkowych.
10. Procesory tekstu, arkusze kalkulacyjne i programy do zarządzania wiadomościami e-mail to przykłady programów narzędziowych.

Krótka odpowiedź

1. Dlaczego procesor jest najważniejszym komponentem komputera?
2. Jaką wartość reprezentuje włączony (ustawiony) bit? Jaką wartość reprezentuje wyłączony bit?
3. Jakim przymiotnikiem określisz urządzenie, które w jakiś sposób przetwarza dane w systemie binarnym?
4. Jaką nazwę noszą słowa, z których składa się język programowania wysokiego poziomu?
5. Jak nazywają się krótkie słowa występujące w języku asemblera?
6. Czym się różni kompilator od interpretera?
7. Oprogramowanie którego typu jest odpowiedzialne za sterowanie sprzętem, z jakiego składa się dany komputer?

Ćwiczenia

1. W dodatku D znajdziesz informacje dotyczące zamiany liczb w systemie dziesiętnym na liczby w systemie binarnym. Wykorzystaj technikę przedstawioną w dodatku D i zamień do postaci binarnej następujące liczby:
   11
   65
   100
   255
2. Wykorzystaj wiedzę zdobytą podczas lektury tego rozdziału i zamień na liczby w systemie dziesiętnym następujące liczby binarne:
   1101
   1000
   101011

3. Przyjrzyj się tabeli kodów ASCII zamieszczonej w dodatku A i wskaż numery, jakie odpowiadają literom Twojego imienia.
4. Poszukaj w sieci informacji na temat historii języków programowania BASIC, C++, Java i Python, a następnie odpowiedz na następujące pytania:
   - Kto stworzył dany język?
   - Kiedy powstał dany język?
   - Czy twórcy danego języka stworzyli go w jakimś określonym celu? Jeśli tak, to co było tym celem?

Skorowidz

G

H

I

J

K

L

Ł

M

N

O

P

R

S

S

T

U

W

Z

>>> BEZ DOBREGO PROJEKTU ZAWIEDZIE NAJLEPSZY KOD!

Programowanie to nie tylko umiejętność pisania kodu oraz organizowania go w funkcje, procedury i moduły. Nawet najbardziej finezyjne języki i najnowocześniejsze paradygmaty programowania nie zapewnią sukcesu, jeśli projekt aplikacji jest nieprzemyślany, a jej logice brakuje... logiki. Z drugiej strony poświęcenie odrobiny czasu na stworzenie dobrego projektu sprawi, że praca nad pisaniem kodu będzie łatwiejsza, sama aplikacja będzie pracować lepiej i bardziej niezawodnie, a późniejsze modyfikacje i rozwój oprogramowania przysporzą mniej problemów.

Jeśli planujesz napisać swoją pierwszą aplikację, sięgnij po tę książkę. Aby zrozumieć zawartą w niej treść, nie trzeba znać żadnego języka programowania. Pokazano tu, jak działają programy i jak można z ich wykorzystaniem rozwiązywać konkretne zadania. Wiedza o zasadach projektowania jest przekazywana za pomocą pseudokodu i schematów blokowych. Omówiono zarówno podstawowe zagadnienia, takie jak typy danych, zmienne, funkcje, jak i nieco bardziej zaawansowane: programowanie obiektowe, tworzenie graficznych interfejsów użytkownika i pisanie programów sterowanych zdarzeniami. W tym wydaniu książki wprowadzono wiele poprawek i uzupełnień, dotyczących między innymi języków Java, Python i C++ oraz programowania aplikacji dla urządzeń mobilnych.

W tej książce:

- jak działa komputer, czym są programy i do czego służą dane
- z jakich modułów i struktur składają się programy oraz jak działają funkcje
- w jaki sposób pracuje się na plikach oraz czym jest przetwarzanie tekstu
- czym jest rekurencja i do czego można ją wykorzystać
- czym się różni programowanie proceduralne od obiektowego

Tony Gaddis

jest wielce cenionym wykładowcą, laureatem wielu wyróżnień. Otrzymał między innymi tytuł Nauczyciela Roku na uczelni North Carolina Community College oraz nagrodę Teaching Excellence przyznaną przez National Institute for Staff and Organizational Development. Napisał wiele znakomitych książek do nauki programowania w językach: C++, Java, Visual Basic, C#, Python, Alice. Jest uznawany za prawdziwy talent dydaktyczny.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

ISBN 978-83-283-5565-1

Cena: 129,00 zł

PEARSON
ALWAYS LEARNING