Temat C1. Wprowadzenie do programowania i rozwiązywania problemów z wykorzystaniem komputera

Warto powtórzyć

1. Jakie znasz rodzaje oprogramowania komputerowego?
2. Z jakich programów użytkowych korzystaliśmy? Jakie było ich przeznaczenie?
3. Jakie są zastosowania języków programowania?
4. Z jakich środowisk programowania korzystaliśmy w szkole podstawowej?
5. Na czym polega programowanie w środowiskach poznanych w szkole podstawowej?

1. Sytuacje problemowe

Programy komputerowe umożliwiają rozwiązywanie problemów.

Algorytm A

Uporządkowany i uściślony sposób rozwiązywania danego problemu, zawierający szczegółowy opis wykonywanych czynności w skończonej liczbie kroków.

Program komputerowy (program źródłowy, kod źródłowy) P

Ciąg instrukcji języka programowania, realizujący algorytm.

Gdy mamy do czynienia z **problemem**, szukamy sposobu rozwiązania (**algorytmu**), a następnie dobieramy odpowiednie narzędzie – **program komputerowy** (np. użytkowy lub język programowania).

Najczęściej korzystamy z programów użytkowych (np. arkuszy kalkulacyjnych, edytorów tekstu lub grafiki, programów do odtwarzania muzyki), programów narzędziowych (np. antywirusowych, archiwizujących) i systemu operacyjnego. Nie jest dla nas ważne, w jaki sposób dany program został napisany. Wystarczy, że znamy jego działanie – niezbędne funkcje i możliwości. Zazwyczaj nie mamy w ogóle wglądu do „wnętrza" programu, czyli do tzw. **programu (kodu) źródłowego**.

Najpierw pojawia się problem, dotyczący np. opracowania wyników zawodów sportowych czy ułożenia planu lekcji. Informatycy zapoznają się z problemem i zastanawiają nad sposobem jego rozwiązania – następnie układają lub wybierają gotowy algorytm.

W podręczniku często podajemy przykłady rozwiązywania problemów dotyczących różnych zagadnień, m.in. edycji tekstu (w punkcie 4. z tematu B1 i w punkcie 1. z tematu B2). Problemy i cele do osiągnięcia wyróżniono nawiasami klamrowymi. Po krótkiej analizie problemu opracowujemy rozwiązanie – w podręczniku podane w przykładzie lub wyróżnione ramką z kluczykiem.

Przykład 1. Przykład sytuacji problemowej i algorytmu jej rozwiązania

Jesteś sprzedawcą komputerów. Cena komputera (*cenak*) zależy m.in. od ceny płyty głównej (*cenap*). Jeśli cena płyty wzrośnie o *w*%, to trzeba podnieść cenę komputera. Przygotowując kalkulację ceny, musisz zastanowić się, o jaki procent możesz podnieść cenę, aby nie stracić klienta.

Przyjmiemy następujący algorytm kalkulacji ceny komputera:

a. jeśli *w* mieści się w przedziale ⟨0%; 10%⟩, cena komputera pozostaje bez zmian (zmniejszasz swój zysk, aby zadowolić klienta);
b. jeśli *w* mieści się w przedziale (10%; 20%⟩, cenę sprzedawanego komputera zwiększasz o 5% nowej ceny płyty głównej (przewidujesz, że klient zaakceptuje wzrost ceny komputera);
c. jeśli $w > 20\%$, nie ustalasz ceny tego komputera (szukasz innego producenta płyty głównej) i wyprowadzasz komunikat: *za wysoka cena*.

W życiu codziennym pewne podobieństwo do algorytmów cechuje np. przepisy kulinarne, określające, z jakich składników i w jaki sposób należy przyrządzić potrawę. Przepisów nie można jednak nazwać algorytmami w sensie informatycznym, choćby z racji tego, że pojawiają się w nich określenia nie do końca precyzyjne, np. „gotować przez chwilę". Zależnie od tego, ile ta chwila będzie trwała, możemy otrzymać różne wyniki (np. budyń o różnej konsystencji).

W algorytmach informatycznych dane wejściowe powinny być precyzyjnie określone, a wszystkie operacje – dokładnie opisane. Dla tych samych danych powinniśmy otrzymać te same wyniki. W algorytmach nieinformatycznych wynik może zależeć od danych wejściowych i różnych czynników (np. od czasu gotowania potrawy).

Wymyślony (lub wybrany) algorytm programiści zapisują w postaci możliwej do wykonania przez komputer, czyli w formie programu komputerowego. Korzystają przy tym z wybranego **języka programowania**. Komputer, wykonując program, realizuje dany algorytm. Ten uproszczony schemat postępowania pokazuje rysunek 1.

J **Język programowania**

Specjalny język służący do pisania programów komputerowych. Jest on zbiorem określonych instrukcji i zasad składni.

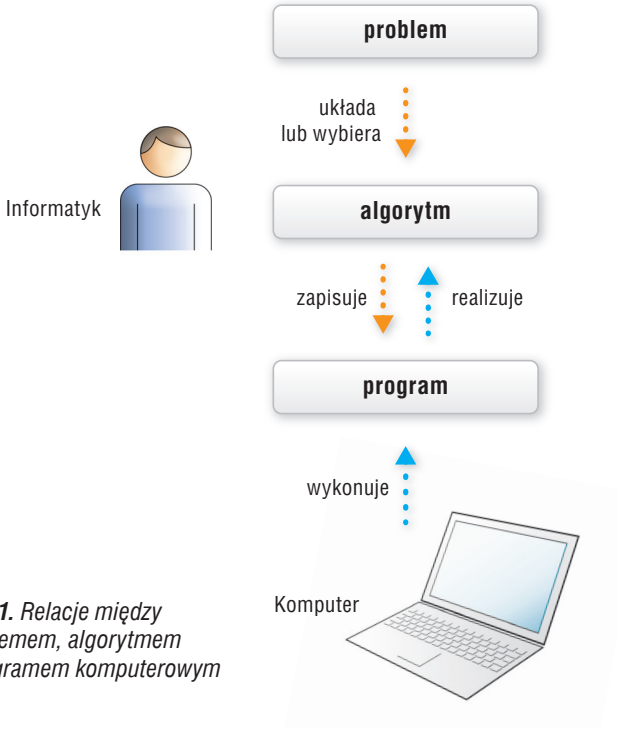
Wybór algorytmu i jego sformułowanie powinny być zawsze podporządkowane problemowi, który ma zostać rozwiązany.

Lista kroków L

Przedstawienie algorytmu w kolejnych punktach (krokach). Każdy punkt takiej listy zawiera opis wykonywanej czynności. Kolejność punktów nie jest przypadkowa – musi być zgodna z działaniem algorytmu.

Schemat blokowy S

Przedstawienie poszczególnych operacji algorytmu za pomocą odpowiednio połączonych figur (bloków). Połączenia określają, w jakiej kolejności i w jaki sposób będą wykonywane operacje realizujące dany algorytm.



Rys. 1. *Relacje między problemem, algorytmem i programem komputerowym*

W rzeczywistości tworzenie programów jest złożonym procesem – tym bardziej, im trudniejszy jest problem. Bardziej skomplikowane zadania dzieli się na mniejsze części i często dla każdej z nich oddzielnie układa bądź dobiera algorytmy. W efekcie powstają duże projekty informatyczne, w których tworzeniu uczestniczy wiele osób, nie tylko programistów, ale również projektantów.

Ćwiczenie 1. Wybieramy algorytm do rozwiązania problemu

Jaki znany ci algorytm można zastosować do znalezienia największego wspólnego dzielnika (NWD) dla dwóch niezerowych liczb naturalnych?

Przykład 2. Przykład sytuacji problemowej

W komputerze gromadzimy wiele danych. Pojawia się zatem problem ich porządkowania, np. w systemach zarządzania plikami: według dat utworzenia, według nazw lub rozszerzeń; w systemach baz danych: sortowanie artykułów według nazw, cen czy dat dostaw.
Aby użytkownik komputera mógł oglądać dane uporządkowane w wybranym przez siebie układzie, informatycy musieli wymyślić odpowiednie sposoby rozwiązania problemu (algorytmy sortowania). Dopiero potem mogli napisać programy do konkretnych zastosowań.

Program komputerowy jest realizacją wybranego wcześniej algorytmu lub wielu algorytmów.

Istniejące programy są usprawniane – pojawiają się ich kolejne wersje, oferujące nowe możliwości, które pozwalają rozwiązywać coraz więcej problemów. Ciągle powstają też nowe programy, dostosowane do zmieniających się potrzeb użytkowników.

Czasem tworzy się programy dla indywidualnych odbiorców (osób prywatnych, firm), jeśli mają oni wymagania użytkowe, których nie spełnia żaden z istniejących programów. Użytkownik może np. oczekiwać innych od oferowanych w gotowym programie sposobów wprowadzania danych (formularzy), odmiennych końcowych zestawień (raportów) czy dodatkowych wyliczeń. Możemy też wyobrazić sobie, że firma użytkownika to swoisty „magazyn osobliwości", zawierający artykuły, które trudno byłoby klasyfikować i przetwarzać według ogólnych kryteriów.

Nierzadko tworzy się też programy do zadań, które wprawdzie można wykonać za pomocą istniejącego oprogramowania, np. arkusza kalkulacyjnego, ale łatwiej i wygodniej jest napisać odrębny program. W nowym programie zadanie będzie wymagało jedynie kilku prostych instrukcji, podczas gdy w arkuszu kalkulacyjnym byłoby bardziej pracochłonne – wymagałoby większej liczby czynności, np. kopiowania tej samej formuły.

Postępowanie mające na celu rozwiązanie dowolnego problemu możemy podzielić na następujące etapy:

1. Sformułowanie zadania.
2. Określenie danych wejściowych.
3. Określenie celu, czyli wyniku.
4. Określenie metody rozwiązania, czyli wybór algorytmu.
5. Przedstawienie algorytmu w wybranej postaci: opisu słownego, listy kroków, schematu blokowego lub programu w wybranym języku programowania.
6. Analiza poprawności rozwiązania.
7. Testowanie rozwiązania dla różnych danych – ocena efektywności przyjętej metody.

! Uwaga

W kolejnych tematach modułu C algorytmy będziemy opisywać i przedstawiać w postaci programów komputerowych, a na lekcjach realizowanych w zakresie rozszerzonym – w postaci list kroków, schematów blokowych i programów komputerowych.

2. Określanie specyfikacji zadania (problemu)

W zadaniach z różnych dziedzin (np. z fizyki czy matematyki) określa się dane wejściowe i wynik (wyniki), a także związek pomiędzy danymi a wynikami, czyli warunki, jakie muszą spełniać wyniki – określamy w ten sposób specyfikację zadania. Potem szukamy odpowiednich rozwiązań, dobieramy wzory, twierdzenia, definicje. Podobnie na zajęciach z informatyki – opis sposobu rozwiązania problemu będziemy poprzedzać **specyfikacją zadania**.

S Specyfikacja zadania (problemu)

Szczegółowy opis zadania, w którym określa się dane wejściowe i oczekiwane wyniki oraz związek między danymi a wynikami (warunki, jakie muszą spełniać wyniki).

Określając specyfikację zadania, warto nadać danym i wynikom nazwy, którymi będziemy się posługiwać w dalszych etapach rozwiązywania zadania.

Przykład 3. Formułowanie zadania i określanie jego specyfikacji – przykłady

Zadanie 1: Oblicz wartość bezwzględną dowolnej liczby całkowitej.
Dane: dowolna liczba całkowita: *x*.
Wynik: wartość bezwzględna liczby *x* równa *w*.

Zadanie 2. Uporządkuj alfabetycznie rosnąco (od A do Ż) zbiór fikcyjnych nazwisk uczniów klas pierwszych fikcyjnej szkoły.
Dane: nieuporządkowany zbiór fikcyjnych nazwisk uczniów klas pierwszych.
Wynik: uporządkowany alfabetycznie zbiór fikcyjnych nazwisk uczniów klas pierwszych.

Zadanie 3. Sprawdź, czy dany wyraz (ciąg znaków składający się z liter) jest palindromem. Palindrom to wyraz, który czyta się tak samo od lewej do prawej i od prawej do lewej, np. *potop, kajak*.
Dane: dowolny wyraz.
Wynik: jeżeli wyraz jest palindromem, określenie „tak”, w przeciwnym wypadku – określenie „nie”.

Ćwiczenie 2. Piszemy specyfikację zadania

Napisz specyfikacje do zadań:
1. Oblicz drogę przebytą w czasie *t* przez pojazd poruszający się ze średnią prędkością *v*.
2. Oblicz sumę liczb ujemnych i sumę liczb dodatnich w *n*-elementowym zbiorze liczb rzeczywistych różnych od zera.
3. Oblicz liczbę znaków różnych od spacji w dowolnym tekście.

Przykład 4. Formułowanie zadania i określanie jego specyfikacji

Sformułujemy zadanie i specyfikację do sytuacji problemowej omówionej w przykładzie 1.

Zadanie. Oblicz i wyprowadź aktualną cenę komputera w zależności od wartości procentowego wzrostu ceny płyty głównej (*w*). Jeśli wartość *w* mieści się w przedziale $\langle 0\%; 10\% \rangle$, cenę komputera pozostaw bez zmian; jeśli mieści się w przedziale $(10\%; 20\% \rangle$, cenę komputera zwiększ o 5% nowej ceny płyty głównej; jeśli $w > 20\%$, wyprowadź komunikat: „za wysoka cena”.
Dane: Cena komputera *cenak*, cena płyty głównej *cenap*, wartość *w* – procent, o jaki wzrosła cena płyty głównej. Przyjmujemy założenia: $cenak > 0$, $cenap > 0$, $w \geq 0$.
Wynik: aktualna wartość ceny komputera *cenak* (jeśli $0 \leq w \leq 20$) lub napis „za wysoka cena” (jeśli $w > 20$).

Program komputerowy może zostać napisany na podstawie listy kroków, schematu blokowego lub samej specyfikacji problemu.

Przykład 5. Projektowanie rozwiązania i wybór narzędzi komputerowych

W przykładzie 1. opisaliśmy przykładowy problem, a w przykładzie 4. sformułowaliśmy zadanie i jego specyfikację, podając dane i wyniki oraz warunki, jakie muszą spełniać dane. W przyjętym rozwiązaniu występują sytuacje warunkowe. Kolejne kroki algorytmu będą zatem realizowane zależnie od spełnienia warunku.
Zaczynamy od sprawdzenia, czy $0 \leq w \leq 10$ (gdzie *w* oznacza procent, o jaki wzrośnie cena płyty głównej). Jeśli warunek jest spełniony, wyprowadzamy wynik: aktualną niezmienioną cenę komputera (*cenak*).

W przeciwnym wypadku sprawdzamy, czy $w \leq 20$. Jeśli warunek jest spełniony, obliczamy aktualną cenę płyty głównej

$$cenap = cenap \cdot (1 + \frac{w}{100})$$

i aktualną cenę komputera według wzoru:

$$cenak = cenak + \frac{5}{100} \cdot cenap$$

i wyprowadzamy wynik: aktualną cenę komputera (*cenak*). W przeciwnym wypadku (czyli dla $w > 20$) wyprowadzamy komunikat „za wysoka cena".

Po opisaniu algorytmu należy sprawdzić, czy wszystkie sformułowania i wartości podane w specyfikacji zadania i opisie algorytmu są wystarczająco precyzyjnie określone, by można było zrealizować rozwiązanie na komputerze.
Do rozwiązania zadania wykorzystamy arkusz kalkulacyjny, ponieważ możemy w nim realizować algorytmy z warunkami za pomocą funkcji JEŻELI. Można również stosować zagnieżdżoną funkcję JEŻELI, w której argumentem funkcji JEŻELI jest wartość będąca wynikiem kolejnej funkcji JEŻELI.

Uwaga: Zadanie sformułowane w przykładzie 4. możemy również rozwiązać w wybranym języku programowania. W temacie C4 zrealizujemy je w językach C++ i Python.

Ćwiczenie 3. Rozwiązywanie zadania za pomocą arkusza kalkulacyjnego

Korzystając z opisu przedstawionego w przykładach 4. i 5., zaprojektuj w arkuszu kalkulacyjnym tabelę do wprowadzania danych. Utwórz formułę obliczającą i wyprowadzającą odpowiedni wynik.

Każde rozwiązanie należy przetestować dla różnych danych. Danymi do zadania rozwiązywanego w ćwiczeniu 3. są liczby, określające odpowiednio: cenę komputera, cenę płyty głównej i procent, o jaki zostanie zmieniona cena płyty głównej. Przyjęliśmy odpowiedni algorytm ustalania ceny (przykład 1.). Określiliśmy też związek między danymi a wynikami (przykład 4.).

Testując rozwiązanie, będziemy oceniać **poprawność** zastosowanego algorytmu.

Algorytm jest **poprawny**, jeśli rozwiązuje problem zgodnie ze specyfikacją (czyli dla poprawnych danych daje poprawne wyniki) oraz dla poprawnych danych zawsze kończy swoje działanie (nie zapętla się).

Ćwiczenie 4. Testowanie rozwiązania

Przetestuj rozwiązanie z ćwiczenia 3. dla różnych danych. Pokaż na przykładach, że dla poprawnych danych (różnych wartości *w*, spełniających podane warunki) otrzymujesz poprawne wyniki.

Na zakończenie pracy nad omawianym problemem przeprowadzamy prezentację rozwiązania zadania i omawiamy zastosowane narzędzie. Odpowiadamy m.in. na pytanie, czy zastosowaliśmy właściwe narzędzie. Wybór arkusza kalkulacyjnego jest właściwy, ponieważ arkusz posiada mechanizmy obliczeń zawierających sytuacje warunkowe (funkcję JEŻELI).

Ćwiczenie 5. Przeprowadzanie prezentacji rozwiązania

Przedstaw i omów rozwiązanie zastosowane w ćwiczeniu 3., korzystając z projektora multimedialnego.

3. Języki programowania

Aby przedstawić algorytm w postaci programu komputerowego, należy zapisać go jako ciąg instrukcji języka programowania. Każda instrukcja odpowiada określonej operacji lub ciągowi operacji.

Słowo kluczowe S

Słowo mające szczególne znaczenie w danym języku programowania, np. oznaczające określony rozkaz, instrukcję lub polecenie.

Każdy język programowania posiada swój zbiór instrukcji, w tym **słowa kluczowe**. Podobnie jak języki naturalne, którymi posługują się ludzie, języki programowania posiadają odpowiednie **zasady składni** oraz właściwe **słownictwo**.

Każdy język naturalny dysponuje własnymi słowami, służącymi określaniu czynności i nazywaniu rzeczy. Podobnie jest z językami programowania.

Instrukcje każdego języka programowania realizują czynności takie, jak: wprowadzanie danych, wyprowadzanie wyników, wykonywanie obliczeń, sprawdzanie warunków czy powtarzanie operacji.

Niezależnie od stosowanego języka programowania należy pamiętać, że:

1. Język programowania jest językiem formalnym, co oznacza, że podlega jednoznacznym regułom. Języki naturalne nie posiadają tej cechy, ponieważ czasem sens zdania zależy od kontekstu, w którym zostało ono użyte, lub od naszej interpretacji.
2. Postać instrukcji, w tym słów kluczowych, musi być bardzo precyzyjna – zgodna z zasadami składni. W programie nie może zabraknąć ani jednego koniecznego znaku (np. dwukropka, przecinka, średnika, nawiasu). Tłumaczenie programu na kod maszynowy połączone jest ze sprawdzaniem poprawności składniowej zapisanych instrukcji. Instrukcje błędnie zapisane nie będą mogły zostać przetłumaczone, a tym samym komputer nie będzie mógł wykonać programu.
3. Kolejność zapisywania instrukcji powinna odpowiadać kolejności operacji realizujących dany algorytm.

Ze względu na **poziom wykonywania programu** języki programowania dzielimy na:

- **języki wysokiego poziomu**, np. Java, C, C++, C# (C Sharp), Python, Visual Basic, Pascal, PHP, JavaScript;
- **języki niskiego poziomu** (wewnętrzne), np. assemblery.

4. Na czym polega programowanie?

Zapisanie algorytmu w postaci ciągu instrukcji języka programowania wysokiego poziomu nazywamy **implementacją**. Powstaje wówczas tzw. **program** (**kod**) **źródłowy**.

Kod źródłowy można zapisać w dowolnym edytorze tekstu, najwygodniej jest jednak skorzystać ze specjalnych edytorów dla programistów lub z edytora wbudowanego w **środowisko programistyczne** danego języka.

Program źródłowy (który jest zapisany jako dokument tekstowy) nie może być wykonywany przez procesor, który potrafi wykonywać jedynie **kod maszynowy**. Pisząc program w języku programowania, przygotowujemy jedynie źródło do utworzenia kodu maszynowego, czyli programu rozumianego przez komputer. Komputer, a dokładniej – procesor, wykonuje program w kodzie maszynowym (w języku wewnętrznym procesora).

Program komputerowy może występować w dwóch postaciach:

- jako **program** (**kod**) **źródłowy** – program zrozumiały dla programisty, zapisany jako ciąg instrukcji tekstowych języka programowania wysokiego poziomu, np. C++ (rys. 4.), Python (rys. 5.),
- jako **program** (**kod**) **wynikowy** (**maszynowy**) – program zrozumiały dla komputera, napisany w języku wewnętrznym komputera w postaci ciągu instrukcji dla procesora (rys. 6.).

Program napisany w języku wysokiego poziomu (np. Python, C++) musi zostać przetłumaczony na język niskiego poziomu (język wewnętrzny komputera). Proces ten nazywamy **translacją**. Może ona przebiegać w formie **kompilacji** lub **interpretacji**, wykonywanych w **translatorze**, czyli programie do tłumaczenia programu na kod maszynowy.

Przykładami języków kompilowanych są: C, C++, C# (C Sharp), Pascal. Przykładami języków interpretowanych są: LOGO, Scratch oraz Python. W zadaniach rozwiązywanych w tym podręczniku będziemy używać języków C++ i Python.

W programowaniu bardzo istotna jest prawidłowa kolejność zapisu poleceń.

I **Implementacja**

Zapisanie algorytmu w postaci kodu źródłowego.

K **Kod maszynowy** (program maszynowy, kod wynikowy, program wynikowy)

Program napisany w języku wewnętrznym (maszynowym), wykonywany przez procesor.

I **Interpretacja**

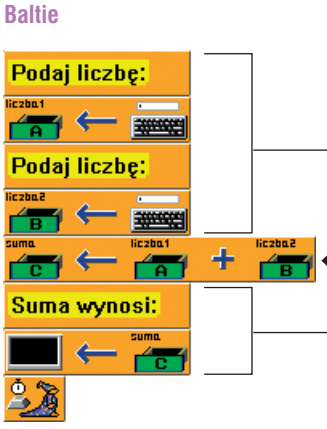
Tłumaczenie programu utworzonego w języku programowania instrukcja po instrukcji, tak aby komputer mógł wykonać każdą z nich. Tłumaczenie następuje każdorazowo przy uruchomieniu programu.

K **Kompilacja**

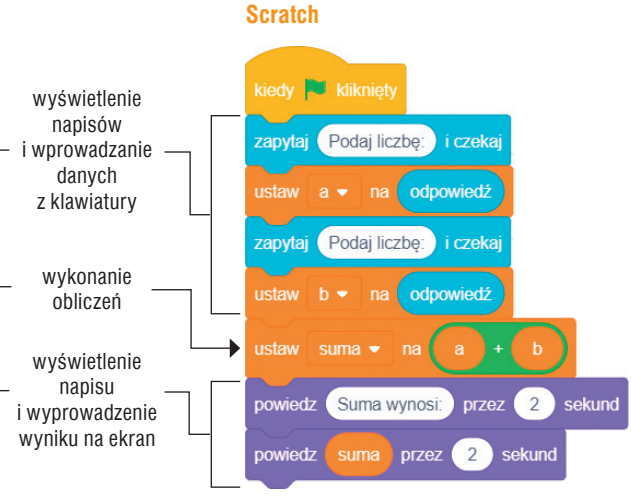
Przetłumaczenie programu w całości na język zrozumiały dla procesora, tak by mógł go wykonać komputer. Jeśli kompilacja przebiegnie poprawnie, można uruchomić program. Raz skompilowany program nie wymaga już powtórnej operacji tłumaczenia.

Przykład 6. Przedstawienie algorytmu obliczenia sumy dwóch liczb w języku programowania

Zadanie: Napisz w wybranym języku programowania program umożliwiający wprowadzenie z klawiatury dwóch liczb całkowitych, obliczający ich sumę i wyprowadzający wynik obliczeń na ekran. Przed wprowadzeniem każdej danej wyświetlaj napis „Podaj liczbę", a przed wyprowadzeniem wyniku – „Suma wynosi:".
Dane: dwie liczby całkowite: *a*, *b*.
Wyniki: wartość sumy liczb *a* i *b*: *suma*.
Na rysunkach 2. i 3. pokazano realizację zadania w środowiskach dydaktycznych Baltie i Scratch, a na rysunkach 4. i 5. – w językach C++ i Python. Na pierwszy rzut oka wydaje się, że są to cztery różne programy. Każde środowisko dydaktyczne ma inaczej wyglądające elementy graficzne, inny zapis poleceń, a każdy z języków programowania może mieć inną postać instrukcji, odmienne zasady składni, w tym stosowanie znaków interpunkcyjnych itp.
Zauważmy jednak, że w każdym rozwiązaniu występują te same części programu, ułożone w takiej samej kolejności:
1. wyświetlenie napisu i wprowadzanie danych z klawiatury,
2. wykonanie obliczeń,
3. wyświetlenie napisu i wyprowadzenie wyniku na ekran.



Rys. 2. Program w środowisku Baltie



Rys. 3. Program w środowisku Scratch

Ćwiczenie 6.

1. Porównaj programy pokazane na rysunkach 2. i 3. – podaj różnice i podobieństwa. Wskaż odpowiadające sobie polecenia i elementy (przedmioty). Na przykład: *Jakiemu elementowi w języku Scratch odpowiada przedmiot z klawiaturą w środowisku Baltie?*
2. Porównaj programy pokazane na rysunkach 4. i 5. – podaj różnice i podobieństwa. Na przykład: Czym różni się zapis instrukcji `suma = a + b` w obydwu językach? Wskaż odpowiadające sobie instrukcje.
3. W wybranym środowisku programowania (Baltie lub Scratch) utwórz program realizujący zadanie opisane w przykładzie 6. Zapisz program w pliku pod nazwą *Suma*.

C++

Rys. 4. Program w języku C++

[*] Suma.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a, b, suma;
    cout << "podaj liczbe: ";
    cin >> a;
    cout << "podaj liczbe: ";
    cin >> b;
    suma = a + b;
    cout << "Suma wynosi: " << suma;
    return 0;
}
```

wyświetlenie napisów i wprowadzanie danych z klawiatury

wykonanie obliczeń

Python

```python
a = int(input("Podaj liczbę: "))
b = int(input("Podaj liczbę: "))

suma = a + b

print("Suma wynosi:", suma)
```

wyświetlenie napisu i wyprowadzenie wyniku na ekran

Rys. 5. Program w języku Python

```
0x00401374      call    0x416610 <__main>
0x00401379      lea     -0x20(%ebp),%eax
0x0040137C      mov     %eax,(%esp)
0x0040137F      movl    $0x1,-0x58(%ebp)
0x00401386      mov     $0x477900,%ecx
0x0040138B      call    0x446984 <std::istream::operator>>(int&)>
0x00401390      sub     $0x4,%esp
0x00401393      lea     -0x24(%ebp),%eax
0x00401396      mov     %eax,(%esp)
0x00401399      mov     $0x477900,%ecx
0x0040139E      call    0x446984 <std::istream::operator>>(int&)>
0x004013A3      sub     $0x4,%esp
```

***Rys. 6.** Program (kod) maszynowy, czyli wynik przetłumaczenia programu napisanego w języku C++ pokazany w symbolicznej formie*

Warto zapamiętać

- Komputery wykonują programy komputerowe, a programy realizują algorytmy (rys. 1.).
- Aby rozwiązać dowolny problem (zadanie), trzeba go poprawnie sformułować, określić dane i wyniki oraz warunki, jakie muszą zostać spełnione. Następnie należy ułożyć lub wybrać odpowiedni algorytm i określić sposób jego realizacji oraz wybrać narzędzie, które to umożliwi.
- Programowanie polega na utworzeniu kodu źródłowego, czyli zapisaniu (implementacji) algorytmu w postaci programu w wybranym języku programowania.
- Aby program był zrozumiały dla komputera, musi zostać przetłumaczony z języka wysokiego poziomu na język wewnętrzny komputera. Powstaje wówczas program (kod) maszynowy (wynikowy).
- Proces tłumaczenia kodu źródłowego na kod maszynowy nazywamy translacją. Może ona przebiegać w formie kompilacji lub interpretacji.

- Wyróżniamy następujące etapy programowania: implementację, kompilację (lub interpretację), uruchomienie i wykonanie oraz testowanie.
- Każdy język programowania posiada odpowiedni zestaw instrukcji, w tym słów kluczowych, umożliwiający zapisanie algorytmów opracowanych przez programistów.

Pytania i polecenia

1. Omów zależności przedstawione na rysunku 1.
2. Wyjaśnij pojęcia algorytmu i specyfikacji problemu (zadania).
3. Jakie są cechy algorytmu informatycznego?
4. Wymień te cechy języka programowania, które odróżniają go od języka naturalnego.
5. Czym jest implementacja?
6. Czym różni się kod źródłowy od kodu maszynowego?
7. Przedstaw i omów na konkretnym przykładzie etapy programowania.
8. Na czym polega kompilacja programu? Czym różni się od interpretacji?

Zadania

1. Napisz specyfikacje do zadań:
   a. Oblicz średnią arytmetyczną trzech liczb.
   b. Określ, czy dana litera jest samogłoską czy spółgłoską. Zależnie od wyniku wyprowadzaj napisy: „samogłoska” lub „spółgłoska”.
   c. Zebrano dane o wzroście uczniów klas pierwszych w twojej szkole. Uporządkuj informacje o wzroście malejąco.
   d. Znajdź wśród danych o wzroście uczniów w twojej klasie najmniejszą i największą liczbę.
2. Wybierz jedno z zadań od 1a do 1d i wykonaj je w arkuszu kalkulacyjnym. Przetestuj rozwiązanie dla różnych danych. Czy arkusz kalkulacyjny jest odpowiednim narzędziem do wykonania każdego zadania? Uzasadnij odpowiedź.
3. W wybranym środowisku dydaktycznym (Baltie lub Scratch) utwórz program realizujący zadanie 1a. Zapisz plik pod nazwą *Średnia.*

Dla zainteresowanych

4. Opisz (w edytorze tekstu lub zeszycie przedmiotowym) przykładową sytuację problemową, w której występują sytuacje warunkowe (możesz wzorować się na opisie problemu z przykładu 1.). Sformułuj zadanie i zapisz jego specyfikację. Zaprojektuj rozwiązanie zadania, wzorując się na przykładzie 5.
5. Wyjaśnij pochodzenie słowa *algorytm*. Informacje znajdź w Internecie.

Przeczytaj, jeśli chcesz wiedzieć więcej…

Pierwszy język wysokiego poziomu powstał w 1954 roku. Nazywał się **FORTRAN** (nazwa pisana wielkimi literami). Kompilator FORTRAN-u był również pierwszym kompilatorem języka wysokiego poziomu. Język ten doczekał się wielu wersji. Do wersji 77. nazywał się FORTRAN (nazwa pisana wielkimi literami), a od wersji 90. do dziś – **Fortran** (nazwa jedynie zaczyna się wielką literą). Co ciekawe, język ten nadal jest rozwijany i używany. Uznaje się go za najpopularniejszy język do obliczeń numerycznych, naukowo-inżynierskich, symulacji. Kolejne wersje umożliwiają programowanie strukturalne, obiektowe, modularne i równoległe.
W Polsce Fortran stosowano w komputerach typu ODRA i Mera.