4.2. Programowanie i techniki algorytmiczne

Warto powtórzyć

1. Czym jest algorytm?
2. Czym jest program komputerowy (kod wynikowy)?
3. Na jakie etapy można podzielić proces rozwiązywania problemu?
4. Czym jest specyfikacja problemu (zadania)?
5. Czym jest lista kroków algorytmu?
6. Na czym polega przedstawienie algorytmu w postaci schematu blokowego?
7. Jakie poznaliśmy rodzaje bloków do graficznego przedstawiania algorytmów?

Na lekcjach informatyki będziemy korzystać z dydaktycznych środowisk programowania: Scratch i Logomocja. Zanim zaczniesz tworzyć programy w tych środowiskach, dowiesz się:
1. na czym polega programowanie,
2. jakie są przykładowe środowiska programowania,
3. czym różni się program źródłowy od programu wynikowego,
4. czym są zmienne w programie,
5. dlaczego stosuje się podprogramy,
6. kiedy mamy do czynienia z sytuacją warunkową,
7. na czym polega iteracja.

1 Na czym polega programowanie?

Aby przedstawić algorytm w postaci programu komputerowego, trzeba zapisać go jako ciąg instrukcji **języka programowania**. Powstaje wówczas tzw. **program (kod) źródłowy**.

Każda instrukcja programu, podobnie jak figura (blok) w schemacie blokowym, odpowiada określonej operacji. Kolejność występowania w programie instrukcji decyduje o kolejności wykonywania operacji.

Aa

Język programowania
Specjalny język służący do pisania programów komputerowych. Jest on zbiorem określonych instrukcji i zasad. Każde słowo lub znak w tym języku jest instrukcją lub jej częścią.

Program komputerowy
(kod źródłowy)
Ciąg instrukcji języka programowania, realizujący algorytm.

Aa

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. Przykładowe języki kompilowane to Pascal i C++.

Interpretacja
Tłumaczenie programu tworzonego w jednym z języków programowania instrukcja po instrukcji, tak by komputer mógł wykonać każdą z nich. Tłumaczenie następuje każdorazowo przy uruchomieniu programu. Przykładem języków interpretowanych są języki edukacyjne Logo i Scratch oraz języki wykorzystywane w tworzeniu stron WWW, np. języki PHP i JavaScript.

Języki programowania możemy podzielić na:
- języki wysokiego poziomu, np. Pascal, C++, JavaScript, Visual Basic;
- języki niskiego poziomu (wewnętrzne), np. assemblery.

Program napisany w języku wysokiego poziomu (kod źródłowy) jest niezrozumiały dla komputera. Komputer potrafi wykonywać tylko instrukcje (rozkazy) zapisane w języku wewnętrznym, zrozumiałym dla procesora (kod wynikowy).

Proces tłumaczenia programu napisanego w języku programowania wysokiego poziomu na język wewnętrzny komputera nazywamy **translacją**. Może on przebiegać w formie **kompilacji** lub **interpretacji**.

Tłumaczeniem kodu źródłowego zajmuje się specjalny program, tzw. **translator**. Tłumaczenie kodu łączy się ze sprawdzaniem poprawności zapisanych instrukcji.

Jeśli instrukcja jest niepoprawna, to komputer jej nie przetłumaczy i wyświetli komunikat o błędzie. Trzeba ją wówczas odszukać, poprawić i powtórnie przeprowadzić translację.

2 Środowiska programowania

Aby tworzyć programy, możemy korzystać z wyspecjalizowanych pakietów programistycznych, zawierających zwykle edytor kodu źródłowego, a także kompilator i inne narzędzia wspomagające programowanie. Przykładami takich środowisk mogą być: Microsoft Visual Studio (które obejmuje m.in. Visual Basic, Visual C# i Visual C++), Delphi, JBuilder.

Korzystając z dydaktycznych środowisk programowania, takich jak Baltie, Scratch i Logomocja, można poznawać podstawowe zasady programowania i tworzyć proste programy. W tych środowiskach upraszcza się sposób wprowadzania poleceń (instrukcji języka programowania). Polecenia mogą być zastępowane m.in. elementami graficznymi, tak jak np. w dostępnych bezpłatnie w internecie programach Baltie i Scratch. Program tworzy się, umieszczając potrzebne elementy w przeznaczonym do tego obszarze. W języku Logo (środowisko Logomocja) wpisujemy polecenia, ale zamiast ich pełnych nazw można używać skrótów.

Do celów dydaktycznych można również wykorzystać język Pascal. Z internetu możemy pobrać bezpłatnie kompilator Free Pascal.

Niezależnie od zastosowanego środowiska programistycznego:

- kolejność występowania instrukcji w programie (także prezentowanych przez elementy graficzne) powinna odpowiadać kolejności operacji realizujących dany algorytm (podobnie jak kolejność bloków w schemacie blokowym);
- postać instrukcji musi być zgodna z zasadami danego języka – nie może zabraknąć nawet jednej spacji, jednego dwukropka, przecinka, średnika czy innego znaku.

3 Program źródłowy i program wynikowy

Na rysunku 1. przedstawiono postać źródłową i wynikową programu zapisanego w języku wysokiego poziomu (w tym przypadku – w języku C++). Program realizuje zadanie: *Napisz program, który umożliwi wprowadzenie dwóch liczb, obliczenie ich sumy i wyprowadzenie wyniku*. Dane do programu i wyniki pamiętane są w oddzielnych zmiennych (*a*, *b* i *s*).

Program wyprowadza wynik działania na ekran, ale równie dobrze mógłby go wydrukować lub zapisać w pliku. Co oznaczają poszczególne wiersze programu w kodzie źródłowym na rysunku 1.?

```
#include <iostream>                {dołączanie biblioteki standardowej wejścia/wyjścia}
using namespace std;               {informacja o korzystaniu z biblioteki standardowej}
int main()                         {początek funkcji głównej programu}
{
     int a, b, s;                  {deklaracja zmiennych}
     cin >> a;                     {wprowadzenie wartości zmiennej a}
     cin >> b;                     {wprowadzenie wartości zmiennej b}
     s = a + b;                    {instrukcja przypisania – wykonanie obliczenia}
     cout << s;                    {wyprowadzenie wyniku}
     return 0;                     {0 oznacza poprawne wykonanie}
}
```

Program w języku wysokiego poziomu (program źródłowy)

```
#include <iostream>

using namespace std;

int main()
{
    int a, b, s;

    cin >> a;
    cin >> b;
    s = a + b ;
    cout << s;
    return 0;
}
```

Wygenerowany kod programu w języku wewnętrznym – fragment (program wynikowy)

```
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
0x004013A6      mov     -0x20(%ebp),%edx
0x004013A9      mov     -0x24(%ebp),%eax
0x004013AC      add     %edx,%eax
0x004013AE      mov     %eax,-0x1c(%ebp)
```

Rys. 1. Postać źródłowa i wynikowa programu

4 Zmienne w programie

Aa

Komórka pamięci
Część pamięci operacyjnej komputera (RAM) o unikatowym adresie.
W komórkach pamięci przechowuje się dane reprezentujące instrukcje procesora lub dane dla tych instrukcji.

Zmiennym wykorzystywanym w programie przyporządkowuje się określone **komórki pamięci**. W wielu językach programowania (np. Pascal i C++) zmienne przed użyciem należy zadeklarować (co w środowiskach Scratch i Logomocja nazywamy tworzeniem zmiennej). Dzięki deklaracji zmiennej kompilator może przydzielić pamięć operacyjną dla zmiennej. W deklaracji zmiennej podaje się jej nazwę oraz typ (określający rodzaj danych przechowywanych w zmiennych – są to np. liczby, znaki, wartości logiczne). Deklaruje się zmienne do przechowywania danych wejściowych dla programu, wyników działania programu, a także danych pomocniczych niezbędnych do wykonania obliczeń.

Zmienna ma zawsze jakąś wartość i określone miejsce w pamięci komputera (adres komórki pamięci). Jeśli w tym samym programie zapiszemy w tej samej zmiennej nową wartość (wykonamy tzw. **instrukcję przypisania**), to poprzednia wartość znika.

Na przykład, po wykonaniu kolejno następujących instrukcji przypisania:

1) `s = 1;`

2) `s = s + 4;`

w zmiennej *s* będzie pamiętana wartość 5.

Gdyby instrukcja 2) była równością w sensie matematycznym, czy byłaby prawdziwa?

5 Dlaczego stosuje się podprogramy?

Kroki, które powtarzają się w algorytmie, można potraktować jako problem cząstkowy i przedstawić w postaci **podprogramu** (**procedury** lub **funkcji**). Umożliwi to opracowanie każdego problemu cząstkowego oddzielnie.

Aa

Podprogram (procedura lub funkcja)
Wyodrębniona część programu, mająca jednoznaczną nazwę i ustalony sposób wymiany danych z innymi częściami programu. Realizujący określone zadanie podprogram można wykorzystywać w programie wielokrotnie.

Aby napisać cały program, wystarczy utworzyć podprogramy i odpowiednio je połączyć. Połączenia podprogramu z programem głównym realizuje się poprzez instrukcję wywołania podprogramu.

Przykłady definiowania podprogramów i ich wywoływania w programie głównym pokażemy w kolejnych tematach, korzystając z dydaktycznych środowisk programowania (Scratch, Logomocja).

6 Sytuacje warunkowe

Z sytuacjami warunkowymi spotykamy się w każdej dziedzinie wiedzy i życia codziennego. Na pytanie „Czy pada deszcz?" odpowiedź może brzmieć „tak" lub „nie". W zależności od tego, czy warunek jest spełniony czy nie, wybieramy sposób postępowania (rys. 2.).

Rys. 2. Przykład sytuacji warunkowej

Ćwiczenie 1. Podajemy przykłady sytuacji warunkowych
Podaj przykłady sytuacji warunkowych z życia codziennego.

Z sytuacją warunkową spotykamy się wówczas, gdy wynik lub dalsze działanie zależy od spełnienia (lub niespełnienia) pewnego warunku. Algorytm, w którym występuje sytuacja warunkowa, będziemy nazywać **algorytmem z warunkami** (inaczej: **algorytmem z rozgałęzieniami**).

W schemacie blokowym sytuacje warunkowe przedstawiamy za pomocą bloku warunkowego. W bloku tym wpisujemy **warunek logiczny**, stosując operatory porównań: = (równy), <> (różny), < (mniejszy), > (większy), <= (mniejszy lub równy), >= (większy lub równy), np. $x >= 0$, $n <> 10$.

Można wpisywać również warunki **złożone**, połączone spójnikami (**i**, **lub**), np. $a < 0$ **lub** $a = 10$, $x > 0$ **i** $x < 20$.

Z bloku warunkowego wychodzą dwa połączenia: TAK i NIE. W zależności od tego, czy warunek jest spełniony czy nie, komputer wybierze odpowiednie odgałęzienie algorytmu.

Z sytuacjami warunkowymi spotykamy się m.in. w matematyce i fizyce, gdy wykonanie obliczeń zależy od warunku, który muszą spełniać liczby (np. mają być nieujemne).

W każdym języku programowania istnieją instrukcje, które umożliwiają zapisanie sytuacji warunkowej.

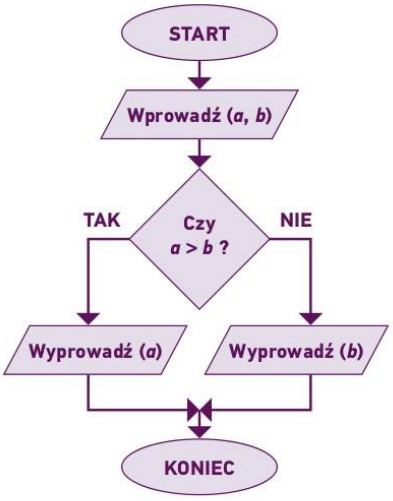
Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**.

Działanie instrukcji warunkowej w większości języków jest podobne. Komputer sprawdza warunek logiczny. W zależności od spełnienia lub niespełnienia warunku wykonuje dalsze instrukcje.

Ćwiczenie 2. Analizujemy schemat blokowy algorytmu z rozgałęzieniami

Na rysunku 3. widać schemat blokowy algorytmu wyboru większej spośród dwóch liczb.

1. Wskaż elementy związane z kolejnymi etapami rozwiązywania zadania: wprowadzanie danych, rozwiązanie zadania (wykonanie różnych operacji) i wyprowadzenie wyników.
2. Jakie rodzaje bloków zastosowano do określenia poszczególnych operacji? Zwróć uwagę na połączenia między blokami. Czy zawsze są takie same?
3. Czy zawsze z jednego bloku „wychodzi" jedno połączenie? Zwróć uwagę na rozgałęzienie w schemacie blokowym.
4. Przetestuj działanie algorytmu dla różnych danych.



Rys. 3. Schemat blokowy algorytmu z warunkami

Ćwiczenie 3. Budujemy schemat blokowy algorytmu z warunkiem prostym

1. Przedstaw w postaci schematu blokowego algorytm obliczania pola P kwadratu o boku a. Uwzględnij w schemacie warunek: dla a dodatniego komputer ma obliczyć pole oraz wyprowadzić wynik, w przeciwnym przypadku algorytm powinien się od razu zakończyć odpowiednim komunikatem, np. „Długość boku kwadratu musi być liczbą dodatnią".
2. Zapisz schemat w pliku pod nazwą *Pole kwadratu*.

Ćwiczenie 4. Budujemy schemat blokowy algorytmu z warunkiem złożonym

1. Przedstaw w postaci schematu blokowego algorytm obliczania pola P trójkąta o podstawie a i wysokości h. Uwzględnij w schemacie warunek: dla a i h dodatnich komputer ma obliczyć pole oraz wyprowadzić wynik, w przeciwnym przypadku algorytm powinien się od razu zakończyć komunikatem: „Wprowadzone dane są nieprawidłowe".
2. Zapisz schemat w pliku pod nazwą *Pole trójkąta*.

Wskazówki:

- Zobacz listę kroków algorytmu obliczania pola trójkąta – temat 4.1, punkt 3.
- Warunek złożony powinien mieć postać: $(a > 0)$ **i** $(h > 0)$.

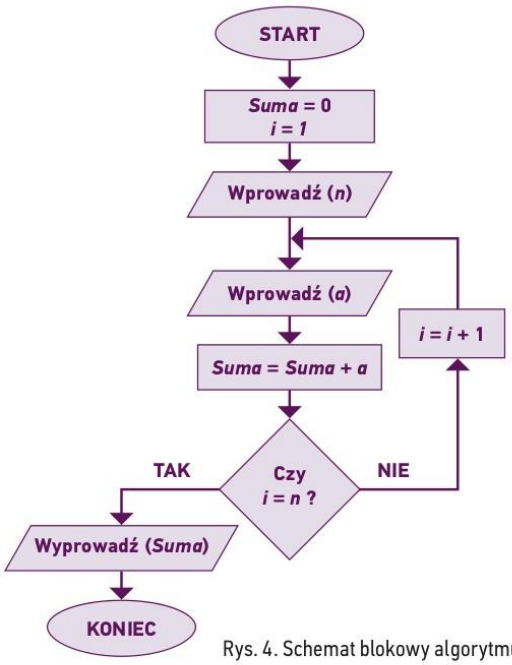
7 Iteracja, czyli powtarzanie poleceń

Iteracja, czyli powtarzanie tych samych operacji, to najczęściej spotykana technika algorytmiczna. Czasami trzeba wykonać te same operacje na wielu danych. W takich przypadkach nie musimy wielokrotnie opisywać takich samych działań lub rysować takich samych bloków.

Iteracja polega na wielokrotnym powtarzaniu tej samej operacji (ciągu operacji). Iterację implementujemy (piszemy kod źródłowy), stosując tzw. pętlę. Z pętlą mamy do czynienia, gdy w pewnym kroku algorytmu wracamy do jednego z wcześniejszych kroków, co powoduje, że kroki te komputer wykona wiele razy.

Ćwiczenie 5. Analizujemy schemat blokowy algorytmu iteracyjnego

1. Schemat blokowy na rysunku 4. jest prezentacją graficzną algorytmu sumowania n liczb, gdzie n jest liczbą naturalną dodatnią. Na początku algorytmu podajemy, ilu liczb zamierzamy użyć do obliczeń. Wartość tę zapisujemy w zmiennej n. Algorytm kończy się, gdy użytkownik wprowadzi tyle liczb, ile wynosi wartość zmiennej n.
2. Zwróć uwagę na wprowadzone w schemacie zmienne pomocnicze *i* oraz *Suma*, a także na przypisane im wartości początkowe.
3. Przetestuj działanie algorytmu dla różnych wartości zmiennych a i n.
4. Odpowiedz na pytania:
   a) Jakie są dane wejściowe do zadania i jaki ma być wynik? Zapisz w zeszycie specyfikację zadania (dane i wynik).
   b) W którym miejscu komputer wykonuje działania w pętli?
   c) Które operacje powtarzają się?
   d) Ile razy będzie realizowana pętla?
   e) Kiedy kończą się działania w pętli?



Rys. 4. Schemat blokowy algorytmu sumowania *n* liczb

Ćwiczenie 6. Stosujemy powtarzanie poleceń

1. Zmodyfikuj schemat wykonany w ćwiczeniu 3. Dla liczby *a* niedodatniej nie kończymy algorytmu, tylko wprowadzamy kolejną liczbę.
2. Zastanów się, w które miejsce schematu powinno wracać połączenie od bloku warunkowego.
3. Jaki jest warunek zakończenia działań?

W każdym języku programowania istnieją instrukcje, które służą do zapisania iteracji. W niektórych językach istnieje kilka takich instrukcji.

Aby w języku programowania napisać program realizujący algorytm iteracyjny, stosujemy **instrukcje iteracyjne**.

Działanie instrukcji iteracyjnych w większości języków jest podobne. Często występuje instrukcja, w której należy z góry określić liczbę powtórzeń, ale są też instrukcje, w których liczba powtórzeń zależy od warunku.

Instrukcje, które chcemy powtórzyć, zapisujemy w sposób określony w danym języku, np. w nawiasach ([...], {...}).

W realizacji algorytmów iteracyjnych ważne jest prawidłowe określenie sposobu zakończenia działań. Niepoprawne określenie tych warunków może spowodować, że obliczenia nigdy się nie zakończą, czyli nastąpi zapętlenie algorytmu.

Zapamiętaj

- Programowanie polega na przedstawieniu algorytmu w postaci instrukcji języka programowania, w kolejności wyznaczonej przez ten algorytm.
- Komputer wykonuje tylko instrukcje (rozkazy) zapisane w języku wewnętrznym, zrozumiałym dla procesora.
- Aby komputer mógł wykonać program, musimy go przetłumaczyć z języka wysokiego poziomu na język wewnętrzny komputera. Proces tłumaczenia nazywamy translacją, która może przebiegać w formie kompilacji lub interpretacji.
- Do tworzenia programów możemy stosować dydaktyczne środowiska programowania lub wyspecjalizowane pakiety programistyczne.
- Zmiennym wykorzystywanym w programie komputer przyporządkowuje określone komórki pamięci operacyjnej o unikatowym adresie. Poza miejscem w pamięci ze zmienną związana jest zawsze jakaś wartość.
- Problem możemy podzielić na problemy cząstkowe, a następnie każdy z nich przedstawić w postaci oddzielnego podprogramu (procedury lub funkcji).
- W algorytmie z warunkami występują sytuacje warunkowe – wynik lub dalsze działanie algorytmu zależy od spełnienia określonego warunku.
- W schemacie blokowym sytuacje warunkowe przedstawiamy za pomocą bloku warunkowego, w którym wpisujemy warunek logiczny. Z bloku warunkowego wychodzą dwa połączenia: TAK i NIE (prawda i fałsz).
- Technikę iteracji stosuje się do czynności wielokrotnie powtarzanych. Dzięki zastosowaniu pętli możemy wrócić do jednego z wcześniejszych kroków algorytmu.
- Zakończenie działań w pętli może zależeć od zadanej liczby powtórzeń.

Serwis dla ciekawskich

Nazwa języka programowania Ada (który powstał w latach 70. XX wieku) pochodzi od imienia kobiety, którą wielu uważa za pierwszą programistkę komputerów. W pierwszej połowie XIX wieku Ada Lovelace (czyt. ejda lawlejs) współpracowała z Charlesem Babbage'em (czyt. czarlsem babydżem) przy projektowaniu pierwszej programowalnej maszyny liczącej (której jednak nigdy nie skonstruowano...). Tworzone przez lady Lovelace opisy rozwiązywania konkretnych zadań obliczeniowych uznaje się za pierwsze programy.