4.1. Sposoby przedstawiania algorytmów

Warto powtórzyć

1. Podaj przykładowe zadanie tekstowe z matematyki. Wskaż dane do zadania, szukane oraz metodę rozwiązania.
2. Wymień etapy tworzenia projektu grupowego.

Na lekcjach matematyki i fizyki większość zadań rozwiązujemy według pewnych schematów: wypisujemy dane, precyzujemy, które wartości należy obliczyć, czyli określamy szukane, wybieramy metodę rozwiązania, wypisujemy wzory łączące dane z szukanymi, przeprowadzamy obliczenia, formułujemy odpowiedź. Rozwiązanie wybranego zadania staramy się opisać, używając wzorów, symboli oraz komentarzy słownych. Rozwiązując zadania na lekcjach informatyki, postępujemy podobnie. W tym temacie dowiesz się:

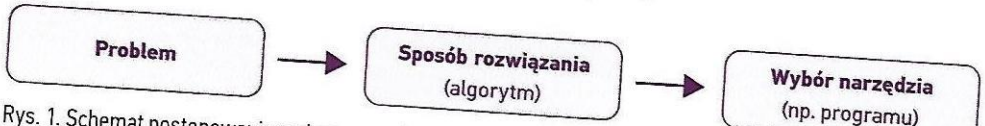
1. na czym polega rozwiązywanie problemów,
2. jakie są etapy rozwiązywania problemów,
3. jak przedstawia się algorytm w postaci listy kroków,
4. jak przedstawia się algorytm w postaci schematu blokowego.

1 Na czym polega rozwiązywanie problemów?

Aa

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

Aby rozwiązać dowolny problem (zadanie), nie tylko z informatyki, należy rozpocząć od poprawnego **sformułowania problemu** oraz ustalenia **danych** i warunków, które te dane muszą spełniać, np. czy ma to być tekst, data, liczba (jej format). Należy też określić cel, czyli **wyniki** (szukane), które chcemy otrzymać, a także warunki, jakie powinny one spełniać. Następnie należy zastanowić się nad sposobem rozwiązania, czyli **algorytmem**, a także wyborem odpowiedniego narzędzia, np. programu komputerowego (rys. 1.).



Rys. 1. Schemat postępowania podczas rozwiązywania problemów za pomocą komputera

Opisanie zadania, czyli przedstawienie związku, który zachodzi między danymi a wynikami, nazywamy **specyfikacją zadania**.

Algorytmy można przedstawiać w postaci: opisu słownego, listy kroków, schematu blokowego, programu w wybranym języku programowania.

2 Etapy rozwiązywania problemów

W procesie rozwiązywania dowolnego zadania, nie tylko za pomocą komputera, można wyróżnić kilka etapów:

1. Sformułowanie zadania.
2. Określenie danych wejściowych.
3. Ustalenie celu, czyli wyniku.
4. Określenie metody rozwiązania, czyli wybór algorytmu.
5. Przedstawienie algorytmu w wybranej postaci.
6. Analiza poprawności rozwiązania.
7. Testowanie rozwiązania dla różnych danych – ocena efektywności przyjętej metody.

3 Lista kroków algorytmu

Sposób rozwiązania zadania, czyli algorytm, można przedstawić w punktach. Mówimy wówczas, że algorytm jest zapisany w postaci **listy kroków**. Operacje powinny być opisywane zgodnie z kolejnością ich realizacji w danym algorytmie. Sposób rozwiązania zadania można również omówić (bez zapisywania go).

Lista kroków to przedstawienie algorytmu w kolejnych punktach (krokach). Każdy punkt takiej listy zawiera opis wykonywanej czynności. Kolejność punktów nie może być przypadkowa – musi być zgodna z działaniem algorytmu.

Przykładem zapisu algorytmu w postaci listy kroków jest przepis na przygotowanie kisielu.

Lista kroków:

1. Z 0,5 litra zimnej wody odlać połowę.
2. Wsypać do niej zawartość torebki.
3. Dobrze wymieszać.
4. Pozostałą część wody zagotować.
5. Dodać trzy łyżki cukru.
6. Do wrzątku wlać rozrobiony proszek.
7. Gotować przez chwilę.
8. Kisiel wlać do salaterek wypłukanych zimną wodą.
9. Podawać gorący lub wystudzony – z konfiturami, sokiem lub śmietaną.

W podanej liście kroków niektóre sformułowania są mało precyzyjne, np. w punkcie 7.: „gotować przez chwilę". Zależnie od tego, ile ta chwila będzie trwała, możemy otrzymać różne wyniki (np. kisiel 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).

Przykład specyfikacji zadania i listy kroków algorytmu

Zadanie: Przedstaw w postaci listy kroków algorytm obliczania pola trójkąta o podstawie a i wysokości h.
Dane: Dowolne liczby rzeczywiste dodatnie: a, h (a – długość boku trójkąta, h – długość wysokości trójkąta opuszczonej na ten bok).
Wynik: Wartość pola trójkąta: P.
1. Zacznij algorytm.
2. Wprowadź wartości liczb a i h.
3. Zmiennej P przypisz wartość wyrażenia: $P = a \cdot h / 2$.
4. Wyprowadź wynik: P.
5. Zakończ algorytm.

Użyte w kroku 3. słowo „przypisz" zapożyczono z języków programowania. Na przykład w języku C++ zapis: $P = a \cdot h / 2$ to **instrukcja przypisania**, w której zmiennej po lewej stronie (czyli P) przypisuje się wartość wyrażenia znajdującego się po prawej stronie instrukcji.

Ćwiczenie 1. Zapisujemy algorytm w postaci listy kroków
Zapisz specyfikację zadania i listę kroków algorytmu obliczania pola trapezu.
Wskazówka: Wzór na pole trapezu: $P = (a + b) \cdot h / 2$, gdzie a, b to długości podstaw trapezu, h – długość wysokości trapezu.

4 Budowanie schematu blokowego przedstawiającego prosty algorytm

W schemacie blokowym poszczególne operacje przedstawia się za pomocą odpowiednio połączonych figur (bloków). Połączenia określają kolejność i sposób wykonania operacji w danym algorytmie.

W tabeli 1. opisano figury, które stosuje się do budowania schematów blokowych.

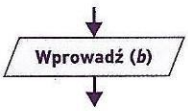
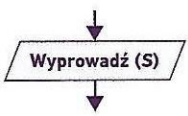
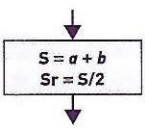
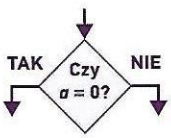
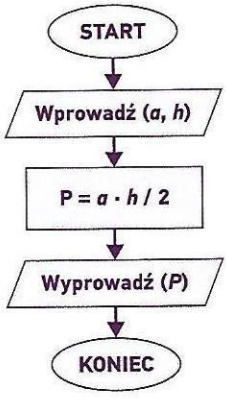
Reprezentacja graficzna	Opis operacji	Uwagi
	**Początek algorytmu**	Blok z napisem „Start" zaczyna algorytm. Wychodzi z niego tylko jedno połączenie i żadne do niego nie wchodzi. W schemacie może funkcjonować tylko jeden taki blok.
	**Zakończenie algorytmu**	Blok z napisem „Koniec" kończy algorytm. Wchodzi do niego jedno połączenie, żadne nie wychodzi. W schemacie może być wiele takich bloków.
	**Wprowadzanie danych** (*blok wejścia*)	Blok z napisem „Wprowadź" służy do wprowadzania danych. Ma jedno połączenie wchodzące i jedno wychodzące. W schemacie może być wiele takich bloków.
	**Wyprowadzanie wyników** (*blok wyjścia*)	Blok z napisem „Wyprowadź" służy do wyprowadzania wyników. Ma jedno połączenie wchodzące i jedno wychodzące. W schemacie może być wiele takich bloków.
	**Wykonywanie działań** (*blok operacyjny*)	Blok, w którym wykonywane są różne operacje, m.in. obliczenia. Ma jedno połączenie wchodzące i jedno wychodzące. W jednym bloku można wpisać więcej niż jedno wyrażenie. W schemacie może być wiele takich bloków.
	**Sprawdzanie warunku** (*blok warunkowy albo decyzyjny*)	Blok podejmowania decyzji. Wchodzi do niego jedno połączenie, wychodzą dwa: • z napisem „Tak", gdy warunek jest spełniony; • z napisem „Nie", gdy warunek nie jest spełniony. W schemacie może być wiele takich bloków.
	**Łącznik**	Łącznik stosuje się, gdy schemat blokowy rysujemy w kilku częściach, np. na dwóch stronach. Umieszczony wewnątrz numer powinien być taki sam w obu łączonych częściach.
	**Połączenie**	Połączenie łączy bloki. Tworzy je linia prosta bądź łamana, zakończona strzałką. Połączenie może dochodzić również do innego połączenia.

Tabela 1. Figury geometryczne stosowane w graficznym przedstawianiu algorytmów

Zasady przedstawiania algorytmów w postaci schematu blokowego

1. Operacje algorytmu należy umieszczać w odpowiednich blokach.
2. Każdy schemat blokowy ma jeden blok startowy, natomiast bloków zakończenia algorytmu może być kilka.
3. Wszystkie bloki muszą być ze sobą połączone (nie może być przerw w schemacie).
4. Każde połączenie wychodzi z danego bloku i dochodzi do następnego bloku lub innego połączenia.
5. Kolejność wykonywania operacji wyznaczają połączenia między blokami.
6. Do każdego bloku wchodzi jedno połączenie (oprócz bloku początku algorytmu) i jedno połączenie z niego wychodzi (oprócz bloku warunku, z którego wychodzą dwa połączenia, oraz bloku zakończenia algorytmu, z którego nie wychodzi żadne połączenie).

Ćwiczenie 2. Sprawdzamy działanie algorytmu
Przetestuj działanie algorytmu pokazanego na rysunku 2. dla następujących liczb: (17; 9), (5; 10), (24; 13).



Rys. 2. Schemat blokowy algorytmu obliczania pola trójkąta

Do rysowania schematów blokowych można wykorzystać **Kształty** (**Autokształty**), dostępne w programach: Microsoft Word, Microsoft PowerPoint lub Paint. Można również poszukać w internecie darmowych programów, służących do tworzenia schematów blokowych.

Ćwiczenie 3. Budujemy schemat blokowy
Narysuj schemat blokowy algorytmu na podstawie listy kroków utworzonej w ćwiczeniu 1.
Wskazówka: Jeśli wykonujesz zadanie w edytorze tekstu, korzystając z **Kształtów**, pamiętaj o możliwości grupowania wstawionych obiektów.

Ćwiczenie 4. Przedstawiamy algorytm w postaci listy kroków i schematu blokowego
Dane są dwie liczby: a i b. Przedstaw w postaci listy kroków i schematu blokowego algorytm obliczania ich sumy oraz wyprowadź jej wartość.

Zapamiętaj

- Algorytm przedstawia krok po kroku sposób rozwiązania problemu.
- Rozwiązując dowolny problem, postępujemy według podobnego schematu. Określamy specyfikację zadania: dane wejściowe, poszukiwane wyniki oraz sposób rozwiązania.
- Poznaliśmy dwa sposoby zapisywania algorytmów: w postaci listy kroków i schematu blokowego.
- Zapis algorytmu w postaci listy kroków polega na przedstawieniu algorytmu w kolejnych punktach (krokach). Każdy punkt takiej listy zawiera opis wykonywanej czynności.
- W schemacie blokowym, czyli graficznej prezentacji algorytmu, układ bloków i połączeń między nimi określa kolejność i sposób wykonywania kroków algorytmu.

Serwis dla ciekawskich

Angielski matematyk Alan Turing opracował w 1936 roku model teoretyczny maszyny do wykonywania algorytmów. Maszyna składała się z nieskończenie długiej taśmy, zawierającej komórki z przetwarzanymi symbolami, głowicy odczytująco-zapisującej i układu sterowania nią. Obliczenia wykonywane za pomocą tej maszyny zależały od układu symboli wpisanych na taśmie oraz od przyjętego zestawu instrukcji.

Podobnie działają dzisiejsze komputery – wyniki obliczeń zależą od danych zapisanych w pamięci komputera i od zestawu wykonanych instrukcji.

OKO W OKO Z MONITOREM

Pytania i polecenia

1. Co to jest algorytm?
2. Czym jest specyfikacja zadania?
3. Wymień kolejne etapy rozwiązywania problemów.
4. Czy przepis na kisiel ma cechy algorytmu informatycznego? Uzasadnij odpowiedź.
5. Wymień cztery sposoby przedstawiania algorytmów.
6. Na czym polega zapisywanie algorytmu w postaci listy kroków?
7. Jakiego bloku użyjesz w schemacie blokowym do wyprowadzenia wyników obliczeń? Uzasadnij odpowiedź.
8. Czy do każdej figury w schemacie blokowych musi dochodzić jedno połączenie? Uzasadnij odpowiedź.
9. W jaki sposób dodaje się dwa ułamki zwykłe o różnych mianownikach? Przygotuj słowny opis tego algorytmu.

Zadania

1. Przedstaw w postaci listy kroków algorytm gotowania budyniu.
2. Przedstaw w postaci listy kroków algorytm obliczania objętości prostopadłościanu.
3. Przedstaw w postaci schematu blokowego algorytm z zadania 2.
4. Przedstaw w postaci schematu blokowego algorytm rozwiązania następującego zadania: *Znajdź cenę 1 m^2 łąki o powierzchni 80 arów, za którą zapłacono 56 000 zł.*
5. Przedstaw w postaci schematu blokowego algorytm obliczania objętości ostrosłupa o wysokości h. Podstawą ostrosłupa jest trójkąt prostokątny o przyprostokątnych a i b.

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

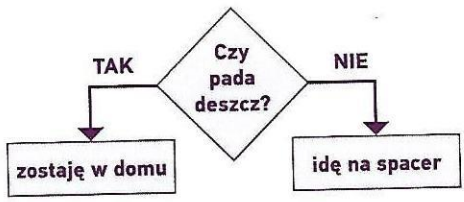
Podprogram (procedura lub funkcja)
Wyodrębniona część programu, mająca jednoznaczną nazwę i ustalony sposób wymiany danych z innymi częściami programu. Realizujący określone zadanie podprogram można wykorzystywać w programie wielokrotnie.

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.

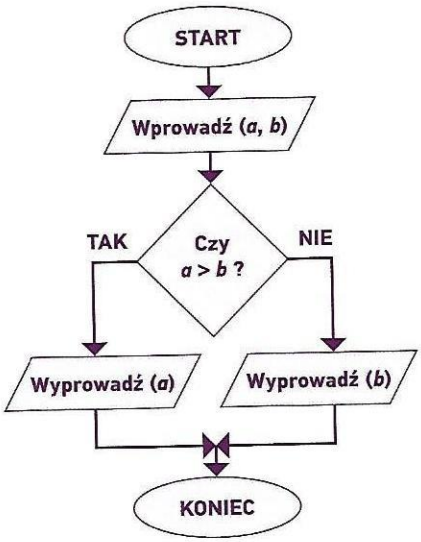
Aby w języku programowania napisać program realizujący algorytm z warunkami, stosujemy **instrukcję warunkową**.

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** $(h > 0)$.

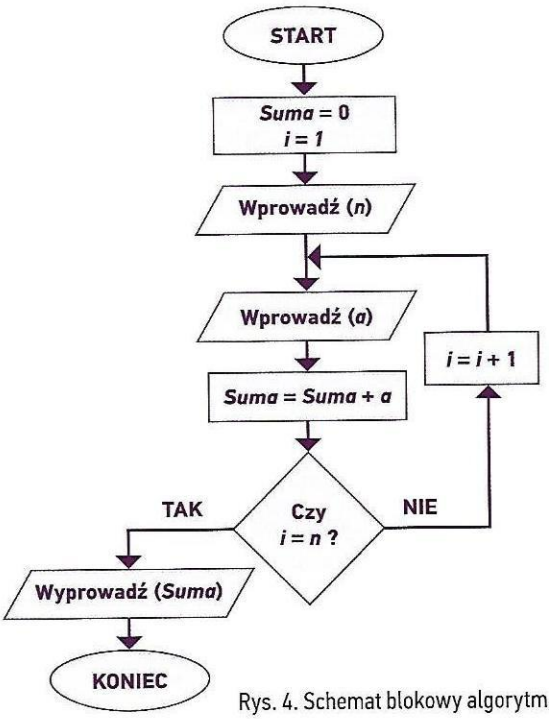
7 Iteracja, czyli powtarzanie poleceń

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.

OKO W OKO Z MONITOREM

Pytania i polecenia

1. Na czym polega programowanie?
2. Czym jest język programowania?
3. Wskaż różnice między kompilacją a interpretacją programu.
4. Jakich narzędzi można użyć do tworzenia programów?
5. Czym się różni postać źródłowa od postaci wynikowej programu? Pokaż na przykładzie.
6. Na co pozwala kompilatorowi deklaracja zmiennej?
7. Czy stosowanie podprogramów jest przydatne? Uzasadnij odpowiedź.
8. Podaj dwa przykłady sytuacji warunkowych, z którymi najczęściej spotykasz się w życiu codziennym.
9. Dlaczego z bloku warunkowego w schemacie blokowym powinny wychodzić dwa połączenia, a tylko jedno wchodzić? Wyjaśnij na przykładzie.
10. W jaki sposób za pomocą schematu blokowego przedstawiamy wielokrotnie powtarzające się działania?
11. W jaki sposób można określić zakończenie iteracji?
12. Jakie instrukcje pozwalają na zapisywanie w programach komputerowych algorytmów z warunkami, a jakie – iteracyjnych?

Zadania

1. Przedstaw w postaci listy kroków algorytm przejścia przez ruchliwe skrzyżowanie (bez sygnalizacji świetlnej).
2. Przedstaw w postaci schematu blokowego algorytm wyznaczania wartości bezwzględnej dowolnej liczby rzeczywistej x.

Wskazówki:

- Wartością bezwzględną liczby nieujemnej jest ta sama liczba, a wartością bezwzględną liczby ujemnej jest liczba do niej przeciwna.
- W bloku warunkowym wpisz $x >= 0$.
- W blokach wykonania wpisz odpowiednio: $w = x$ (gdy warunek jest spełniony), $w = -x$ (gdy warunek nie jest spełniony).
- Wyprowadź wynik w.

3. Przedstaw w postaci schematu blokowego następujący algorytm: wprowadzamy dwie liczby a i b (zakładamy, że są to zawsze liczby dodatnie i różne). Dla liczby większej oblicz objętość sześcianu o krawędzi długości równej tej liczbie. Dla liczby mniejszej oblicz pole kwadratu o boku długości równej tej liczbie.

4. Przedstaw w postaci schematu blokowego algorytm obliczania pola powierzchni sześcianu o krawędzi a. Uwzględnij w schemacie warunek: dla a dodatniego komputer ma obliczyć pole i wyprowadzić wynik, w przeciwnym przypadku algorytm powinien się od razu zakończyć odpowiednim komunikatem.

5. Przedstaw w postaci schematu blokowego algorytm obliczania objętości prostopadłościanu o krawędziach a, b, c. Uwzględnij w schemacie warunek: dla a, b, c dodatnich obliczaj objętość i wyprowadź wynik, w przeciwnym przypadku zakończ algorytm odpowiednim komunikatem.

6. Oblicz, o ile procent twój kolega jest wyższy od ciebie. Przedstaw algorytm rozwiązania tego zadania w postaci schematu blokowego. Przyjmij oznaczenia: wzrost kolegi – x, twój wzrost – y, wynik – p. Dla $x > y$ mają być wykonane obliczenia i algorytm ma się zakończyć. W przeciwnym wypadku (dla $x <= y$) należy wprowadzić nowe dane (zastosuj pętlę).

DLA ZAINTERESOWANYCH

1. Przedstaw w postaci schematu blokowego algorytm dodawania n kolejnych liczb naturalnych.

2. Przedstaw w postaci schematu blokowego algorytm sprawdzania, czy istnieje trójkąt o bokach, których długości są równe wprowadzanym liczbom. Na wejściu wprowadzane są trzy liczby.

   Wskazówka: Przypomnij sobie z matematyki warunek istnienia trójkąta.

3. Podaj przykład zadania z fizyki, w którego rozwiązaniu występuje sytuacja warunkowa (lub sytuacje warunkowe). Zapisz algorytm w postaci schematu blokowego i sprawdź jego działanie dla różnych danych.