

Podstawy programowania

1.1. Zasady programowania

Język programowania służy do tworzenia programów komputerowych, których zadaniem jest przetwarzanie danych, wykonywanie obliczeń i algorytmów. Może zawierać konstrukcje składniowe do manipulowania strukturami danych i zarządzania przepływem sterowania. Niektóre języki programowania mają specyfikację swojej składni i semantyki, inne zdefiniowane są jedynie przez oficjalne implementacje.

Elementy języka:

- **Składnia** — zbiór reguł opisujących sposób definiowania struktur danych, rodzaje dostępnych słów kluczowych i symboli oraz zasady, według których symbole mogą być łączone w większe struktury.
- **Semantyka** — zbiór reguł definiujących znaczenie słów kluczowych i symboli oraz ich funkcji w programie.
- **Typy danych** — dostępne typy danych, ich właściwości oraz operacje, które mogą być wykonane na wartościach danego typu.

Przykładami używanych obecnie języków programowania są: C, C++, C#, Java, PHP, Perl, Python, Ruby.

Niektóre z tych języków wywodzą się z języka C, na przykład C++ czy C#. Języki Java i Python są popularne, gdyż pozwalają na bardzo szybkie tworzenie aplikacji. Wydajniejsze języki, takie jak C czy C++, posłużyły do napisania programów biurowych typu edytory tekstu czy edytory grafiki. Języki PHP i Java są popularne w aplikacjach internetowych. Python bywa wykorzystywany w administrowaniu systemami i do tworzenia aplikacji.

Wybór języka programowania może zależeć od przeznaczenia końcowej aplikacji, indywidualnych upodobań lub strategii firmy tworzącej oprogramowanie. Najlepszym

wyjściem jest wybór języka programowania najbardziej dostosowanego do rozwiązywanego zadania i istniejącej infrastruktury.

Kod źródłowy programu jest tworzony i zapisywany w formacie tekstowym. Rozszerzenie pliku zależy od wybranego języka programowania. Identyfikuje ono język, w którym program został napisany, na przykład *c* dla języka C, *cpp* dla C++, *java* dla Javy, *cs* dla C#. Niezależnie od rozszerzenia plik zawierający kod źródłowy programu można otworzyć w dowolnym edytorze tekstu.

1.1.1. Paradygmaty programowania

Paradygmat programowania to pewien wzorzec określający sposób pisania i wykonania programu komputerowego. Języki programowania korzystają z różnych paradygmatów. Paradygmaty programowania opisują między innymi programowanie:

- strukturalne,
- obiektowe,
- proceduralne,
- funkcyjne,
- uogólnione.

Różnice polegają na różnych strukturach programów oraz na różnym podejściu do problemu.

Najbardziej rozpowszechniony jest paradygmat strukturalny (programowanie strukturalne) oraz obiektowy (programowanie zorientowane obiektowo).

W **programowaniu strukturalnym** następuje dzielenie kodu na bloki (procedury i funkcje). Instrukcje są pobierane ze stosu i są wykonywane z wykorzystaniem pętli i instrukcji warunkowych. W większości języków można programować strukturalnie.

W **programowaniu obiektowym** programy definiowane są za pomocą obiektów. Program napisany zgodnie z zasadami programowania obiektowego składa się ze zbioru obiektów, które komunikują się między sobą, aby wykonać określone zadania.

W **programowaniu proceduralnym** program dzielony jest na procedury, czyli fragmenty wykonujące ściśle określone operacje. Procedury nie powinny korzystać ze zmiennych globalnych, lecz wszystkie dane powinny być pobierane i przekazywane jako parametry wywołania.

W **programowaniu uogólnionym** kod programu powstaje bez wcześniejszej znajomości typów danych, na których będzie pracował. Do języków programowania wykorzystujących uogólnienia należą: C++, Java.

Określony język może wspierać różne paradygmaty programowania. Może przykładowo zawierać elementy programowania proceduralnego, obiektowego i uogólnionego. Wtedy jest to **język hybrydowy**. Można w takim języku programować, stosując się na przykład wyłącznie do paradygmatów programowania strukturalnego lub wykorzystując elementy wielu paradygmatów. To programista decyduje, w jaki sposób będzie tworzył program.

1.1.2. Programowanie strukturalne

W programowaniu strukturalnym kod programu jest dzielony na hierarchicznie ułożone bloki. Program tworzony jest z wcześniej zdefiniowanych struktur, takich jak sekwencja (ciąg kolejnych instrukcji), instrukcja warunkowa (`if`, `if...else`), pętla (`while`, `repeat`), instrukcja wyboru. Struktury te są stosowane do małych bloków programu zawierających podstawowe instrukcje typu podstawienia, wejścia-wyjścia lub wywołania procedur i funkcji. Dzięki temu kod jest przejrzysty i łatwy w utrzymaniu.

Występujące w językach strukturalnych instrukcje skoku (`goto`) lub przerywania (`continue`, `switch`) są niezgodne z zasadami strukturalności, ale pojawiają się w językach programowania, ponieważ zwiększają czytelność programu.

Prawie w każdym języku można programować strukturalnie. Jednak niektóre z nich są do tego celu specjalnie przeznaczone. Modelowym przykładem języka strukturalnego był język Pascal.

Reasumując: programowanie strukturalne zachęca do dzielenia programu na bloki (moduły, podprogramy), a powstałych bloków na mniejsze jednostki w kolejności wykonywania. Wykorzystuje instrukcje sterowania i pętle do kontrolowania przebiegu programu. Kod staje się dzięki temu bardziej przejrzysty i wzrasta jego wydajność.

Sposób programowania w tym paradygmacie może być różny w zależności od zastosowanego systemu operacyjnego i języka programowania. Jednak podstawowe zasady programowania powinny zostać niezmiennie.

Zmienna

Jednym z podstawowych elementów języków programowania jest **zmienna**. Jest ona abstrakcją komórki pamięci komputera. Każdą zmienną można opisać za pomocą nazwy, adresu, wartości, typu, okresu życia i zakresu widoczności.

Nazwa — określa nazwę, przez którą następuje odwołanie do zmiennej. Dla nazwy powinny zostać zdefiniowane: zakres dozwolonych znaków, liczba dozwolonych znaków, rozróżnialność małych i dużych liter.

Adres — określa bieżący adres zmiennej w pamięci.

Wartość — określa wartość przypisaną do zmiennej.

Typ — określa zbiór dopuszczalnych wartości, jakie zmienna może przyjmować.

Zakres widoczności — określa blok instrukcji, w których zmienna jest widoczna i można się do niej odwołać.

Okres życia — określa czas, przez który zmienna istnieje. Przykładowo okres życia zmiennej zadeklarowanej w funkcji zaczyna się od wejścia do funkcji i kończy się w momencie zakończenia funkcji. Istnieją również zmienne zadeklarowane w funkcji, które widoczne są tylko w tej funkcji, ale okres życia dotyczy całego czasu wykonania programu.

Proste typy danych

Typ to zbiór wartości, które może przyjmować zmienna. Większość języków programowania ma zestaw typów prostych.

Typ całkowity (*int*, *integer*) odpowiada liczbom całkowitym, ale mogą wystąpić jego warianty związane z rozmiarem liczby (*byte*, *short*, *long*) i jej znakiem (*signed*, *unsigned*).

Typy zmiennopozycyjne odpowiadają liczbom rzeczywistym (*float*, *double*).

Typy znakowe przechowują informacje o znakach zapisane w kodzie ASCII lub coraz częściej w Unicode (*char*, *character*).

Typ logiczny przechowuje wartość logiczną *Prawda* lub *Fałsz* i jest zapisywany za pomocą jednego bitu lub całego bajta.

Typy stałoprzecinkowe mają ustaloną liczbę cyfr i miejsc po przecinku.

Z typów prostych można tworzyć typy złożone, na przykład *rekordy*, *tablice*, *typy wyliczeniowe*.

Operatory

Z każdym typem związana jest grupa operacji, które mogą zostać wykonane na wartościach tego typu. Przykładowo:

Typ całkowity — operacje dodawania (+), odejmowania (−), mnożenia (*), dzielenia (/).

Typ logiczny — operacje koniunkcji (*And*), alternatywy (*Or*), negacji (*Not*).

Typ znakowy — operacja łączenia tekstów (+).

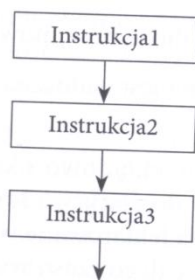
Operacje wejścia-wyjścia

Operacje wejścia-wyjścia pozwalają na pobieranie i wypisywanie danych z konsoli.

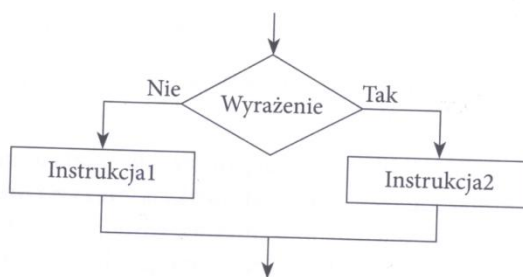
Instrukcje

Sekwencja — ciąg kolejnych poleceń (rysunek 1.1).

Instrukcja warunkowa — pozwala na rozgałęzienie przebiegu działania programu (rysunek 1.2).



Rysunek 1.1. Sekwencja

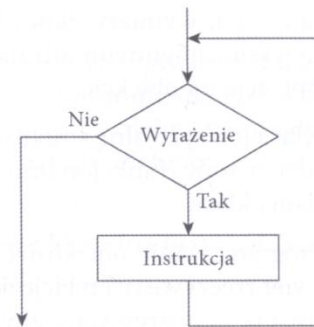


Rysunek 1.2. Instrukcja warunkowa

Jeżeli wartość wyrażenia jest prawdą, to zostanie wykonana *Instrukcja2*, w przeciwnym razie — *Instrukcja1*.

Pętla — pozwala na wykonywanie bloku programu kilkakrotnie w pętli, dopóki wartością sprawdzanego wyrażenia jest prawda (rysunek 1.3).

Dopóki wartość wyrażenia jest prawdą, wykonywana jest *Instrukcja*.



Rysunek 1.3. Pętla

Przykład 1.1

Użycie instrukcji warunkowej w języku C++:

```
#include <iostream>
using namespace std;
int main()
{
    int liczba1 = 30, liczba2 = 16;
    if (liczba1 > liczba2)
        cout << "Liczba: " << liczba1 << " jest większa od liczby " << liczba2;
    else
        cout << "Liczba: " << liczba2 << " jest większa od liczby " << liczba1;
    return 0;
}
```

1.1.3. Programowanie obiektowe

Języki obiektowe w różnym stopniu stosują zasady obiektowości. Popularnymi językami obiektowymi są Java, C# i Python, chociaż pojawiają się w nich elementy proceduralności. Języki C++ i Perl, które pierwotnie były językami programowania strukturalnego, zostały uzupełnione o elementy obiektowości. Niektóre języki, mimo iż nie są językami obiektowymi, pozwalają na stosowanie w ograniczonym zakresie niektórych metod programowania obiektowego (abstrakcyjnych typów danych, dziedziczenia, polimorfizmu). Językami w pełni obiektowymi są C# i Ruby, które wymuszają stosowanie metod programowania obiektowego.

Obiektowość występuje zwykle w systemach hybrydowych w połączeniu z programowaniem sieciowym (Java), skryptowym (Perl, Python, Ruby). Systemy czysto obiektowe, jak Smalltalk, nie są powszechnie stosowane.

W programowaniu obiektowym program to zbiór powiązanych obiektów, które na siebie oddziałują.

Obiekt to element, który jest opisywany przez stan (właściwości) i zachowanie (metody, czyli funkcje). Właściwości obiektu opisują jego cechy (na przykład liczbę znaków

łańcucha, wymiary okna) lub pozwalają określić jego stan. Są też nazywane polami obiektu, zmiennymi lub zmiennymi składowymi. Metody to funkcje, które wykonują operacje na obiekcie.

Klasa jest specjalną strukturą opisującą grupę powiązanych ze sobą obiektów. Definiuje ona metody (funkcjonalność) oraz atrybuty wybranych obiektów. Obiekt jest instancją danej klasy.

Programowanie obiektowe opiera się na tworzeniu programów, które przedstawiają świat rzeczywisty i relacje w nim zachodzące za pomocą obiektów. Program powinien być tak skonstruowany, aby jak najlepiej odzwierciedlał opisywany fragment rzeczywistości. Klasa opisuje za pomocą właściwości (zmiennych) i metod (funkcji) fragment świata rzeczywistego. Właściwości zawierają informacje o stanie tego fragmentu, metody pozwalają na kontrolę i zmianę tego stanu. Obiekt to wystąpienie (konkretna instancja) danej klasy.

Dobrym zwyczajem jest dzielenie kodu źródłowego na klasy ze względu na funkcje. Przykładowo jedna klasa odpowiada za obsługę błędów, inna za wyświetlanie komunikatów.

Wyróżniamy dwa podejścia do programowania obiektowego. Są to:

- programowanie oparte na klasach,
- programowanie oparte na prototypach.

W **programowaniu opartym na klasach** definiowane są klasy, a następnie tworzone są obiekty, które są elementami wybranej klasy (C#).

W **programowaniu opartym na prototypach** nie istnieje pojęcie klasy. Nowe obiekty tworzy się na bazie istniejącego już obiektu (prototypu), po którym dziedziczone są pola i metody (JavaScript).

Cechy obiektowości

Język obiektowy powinien charakteryzować się określonymi cechami. Są to:

- abstrakcja,
- hermetryzacja,
- dziedziczenie,
- polimorfizm.

Abstrakcja

Można powiedzieć, że każde uogólnienie (uproszczenie) rozpatrywanego problemu, które polega na wyodrębnieniu wspólnych cech dla danego zbioru elementów, jest **abstrakcją**. Analizując problem, skupiamy się na ogólnych cechach tego zbioru elementów, a nie na właściwościach poszczególnych elementów.

Hermetryzacja

Hermetryzacja, inaczej **enkapsulacja**, to ukrywanie pewnych właściwości (danych składowych) lub metod obiektów określonej klasy. Dzięki temu są one dostępne tylko w obrębie klasy.

Dziedziczenie

Mechanizm dziedziczenia pozwala na tworzenie klas na podstawie innych wcześniej utworzonych klas. Nowa klasa ma właściwości i metody zdefiniowane dla niej oraz właściwości i metody klasy, na podstawie której została utworzona, zdefiniowane jako *public* i *protected*. Natomiast niedostępne są właściwości i metody zdefiniowane jako *private*.

W językach programowania, w których nie występuje pojęcie klasy (na przykład JavaScript), dziedziczenie zachodzi pomiędzy poszczególnymi obiektami.

Polimorfizm

Słowo **polimorfizm** oznacza wielopostaciowość. W programowaniu obiektowym polimorfizm to możliwość stworzenia obiektu, który ma więcej niż jedną formę. Oznacza to, że możliwe są wystąpienia funkcji o tej samej nazwie, zawierających różne zestawy argumentów i operatorów działających w różny sposób, w zależności od typów argumentów.

1.1.4. Skryptowe języki programowania

Skrypt to napisany w języku skryptowym program, który jest wykonywany wewnątrz aplikacji.

Język skryptowy to język programowania służący do wykonywania wyspecjalizowanych czynności. Języki skryptowe są tworzone z myślą o interakcji z użytkownikiem. Często są wykorzystywane do zadań administracyjnych. Bywają również osadzone w programach w celu zautomatyzowania powtarzających się czynności. Są używane do tworzenia dynamicznych stron internetowych. Stosowane w grach komputerowych służą do sterowania przebiegiem gry.

Języki skryptowe mogą być wykorzystywane do pisania zaawansowanych aplikacji, ale najczęściej są używane do szybkiego tworzenia niewielkich skryptów pozwalających na dynamiczne wyświetlanie stron internetowych lub zapamiętywanie i przetwarzanie wprowadzonych danych.

Języki skryptowe często stanowią dodatkowe narzędzie różnego rodzaju oprogramowania.

Do popularnych języków skryptowych należą:

- JavaScript,
- Python,
- PHP.

JavaScript

JavaScript to skryptowy język programowania stosowany do obsługi stron internetowych. Skrypty działają po stronie użytkownika i najczęściej realizują interakcję

z użytkownikiem z wykorzystaniem obsługi zdarzeń, tworzą efekty animacji na stronie internetowej, obsługują walidację formularzy. Wszystkie popularne przeglądarki internetowe implementują JavaScript i obsługują opracowany przez W3C jednolity model obiektowy reprezentujący drzewo dokumentu (DOM). Umożliwiają również tworzenie plików cookie, wyświetlanie okien dialogowych, manipulowanie oknami przeglądarki.

Python

Python jest językiem programowania wysokiego poziomu. Wspiera różne paradygmaty programowania. Często jest używany jako język skryptowy. Z wykorzystaniem odpowiedniego frameworka (na przykład Flask, Django) może być stosowany do tworzenia aplikacji internetowych.

PHP

PHP to skryptowy język programowania stosowany w aplikacjach internetowych. Służy do tworzenia skryptów po stronie serwera, ale może być również używany do tworzenia aplikacji z interfejsem graficznym i pisania skryptów wykonywanych z wiersza poleceń.

1.1.5. Biblioteki

W tworzonych programach często powtarzane są te same czynności, na przykład wyświetlanie na ekranie, wprowadzanie danych z klawiatury, otwieranie pliku do edycji. Wiele języków programowania dysponuje modułami, które zawierają definicje takich czynności i mogą zostać wykorzystane przez programistę. Są to tak zwane **biblioteki**. Programista może również tworzyć własne biblioteki.

Biblioteki zawierają definicje funkcji najczęściej używanych w programie. Biblioteki rozszerzają zakres operacji dostępnych w danym języku i ułatwiają pisanie programów. Większość języków programowania udostępnia biblioteki standardowe, które zawierają definicje typowych operacji wykonywanych w programach. Należą do nich:

- operacje na ciągach tekstowych,
- operacje na typach danych i funkcje do zarządzania nimi,
- obsługa wejścia-wyjścia,
- obsługa plików,
- obsługa wielowątkowości,
- zarządzanie pamięcią.

Biblioteki mogą być statyczne lub dynamiczne. Biblioteki statyczne są dołączane do programu za pomocą dyrektywy preprocesora na etapie konsolidacji. Biblioteki dynamiczne są dołączane do programu dopiero w czasie jego uruchamiania.

1.2. Proste algorytmy

Algorytm to zestaw ściśle określonych czynności prowadzących do wykonania pewnego zadania. Określa sposób rozwiązywania problemu i ma zastosowanie w różnych dziedzinach. Języki programowania to narzędzia, które bardzo dobrze nadają się do zapisu algorytmów. Aby napisać dobry program komputerowy, należy opracować skuteczny algorytm i zdefiniować dla niego odpowiednie struktury danych.

Algorytm przetwarzania danych powinien przy takim samym zbiorze danych wejściowych zwracać zawsze taki sam wynik. Ale stanie się tak tylko w dokładnie takich samych warunkach i przy tych samych danych pomocniczych. Zwykle przy projektowaniu algorytmu zakłada się, że dane wejściowe są poprawne, ale bywają algorytmy, które nie tylko przetwarzają dane, lecz również je weryfikują.

W rzeczywistości tak jak nie każdy problem można rozwiązać, tak nie każdą metodę rozwiązania problemu można zapisać przy użyciu algorytmu. Aby problem mógł być rozwiązany za pomocą komputera, musi zostać zapisany w postaci algorytmu. Wynika to z tego, że komputer potrafi rozwiązywać tylko problemy, dla których rozwiązanie zostanie zdefiniowane w postaci jednoznacznych kroków, czyli algorytmu. Jeżeli nie można zdefiniować rozwiązania w postaci algorytmu, nie ma możliwości rozwiązania go z wykorzystaniem komputera.

Zdefiniowany algorytm może zostać zapisany w wybranym języku programowania. Ale ten sam algorytm może zostać zapisany różnie w zależności od użytego języka programowania.

Zapis algorytmu w wybranym języku programowania nazywamy **implementacją algorytmu**.

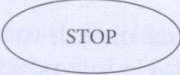
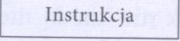
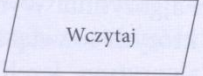
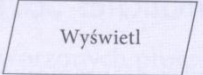
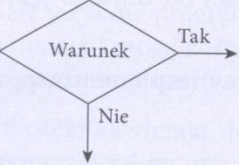
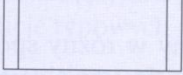
1.2.1. Reprezentacja algorytmów

Algorytm opisujący operacje do wykonania może zostać zapisany w różny sposób. Może to być zapis słowny, lista kroków do wykonania, pseudokod, drzewo algorytmu lub schemat blokowy (tabela 1.1).

Schemat blokowy

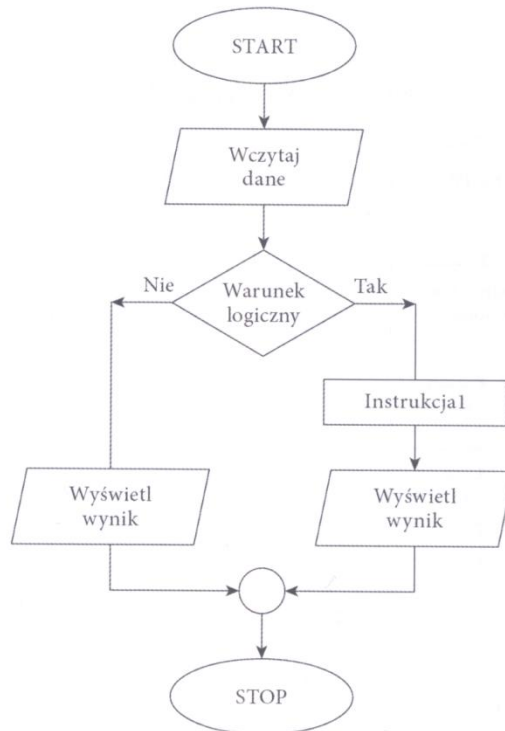
W schemacie blokowym operacje, które należy wykonać, są przedstawiane w postaci graficznej z użyciem symboli.

Tabela 1.1. Symbole wykorzystywane do tworzenia schematów blokowych

Symbol	Opis
	Początek algorytmu, start programu. Od tego miejsca rozpoczyna się wykonywanie operacji.
	Koniec algorytmu, zakończenie programu. W tym miejscu następuje zakończenie wykonywania operacji.
	Połączenie między blokami. Wskazuje kolejność wykonywania operacji.
	Wykonanie operacji, blok obliczeniowy. Wewnątrz tego symbolu znajdują się operacje do wykonania.
	Wprowadzanie danych. Wewnątrz tego symbolu określamy dane wejściowe, które muszą zostać wczytane.
	Wyprowadzanie danych. Wewnątrz tego symbolu określamy dane wyjściowe, które powinny zostać wyprowadzone jako wynik.
	Warunek logiczny, blok decyzyjny. Umożliwia tworzenie rozgałęzień w algorytmie. Jeżeli warunek jest spełniony, to następuje przejście do gałęzi oznaczonej „Tak”, w przeciwnym razie następuje przejście do gałęzi oznaczonej „Nie”.
	Proces wstępnie zdefiniowany. Symbol ten oznacza dołączenie podprogramu.
	Łącznik. Odwołanie na stronie. Służy do oznaczenia miejsc łączenia schematu, na przykład gdyby linie łączące na schemacie musiały się krzyżować.
	Łącznik międzystronicowy. Służy do oznaczenia miejsc łączenia schematu, gdy nie mieści się on na jednej stronie.

Przykład 1.2

Schemat blokowy (rysunek 1.4):



Rysunek 1.4. Przykład schematu blokowego

1.2.2. Przykłady algorytmów

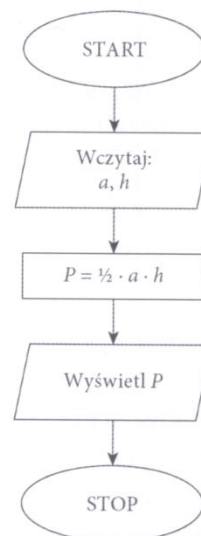
Obliczanie pola trójkąta

Przykład 1.3

Algorytm obliczania pola trójkąta przedstawiony w postaci schematu blokowego (rysunek 1.5).

Sortowanie liczb

Jednym z podstawowych zagadnień algorytmicznych jest porządkowanie zbioru danych według określonych jego cech. Szczególnym przypadkiem porządkowania danych jest sortowanie liczb lub słów. Algorytmy sortowania są klasyfikowane ze względu na sposób działania, złożoność lub stabilność. Prosta metoda sortowania jest sortowanie bąbelkowe. Polega

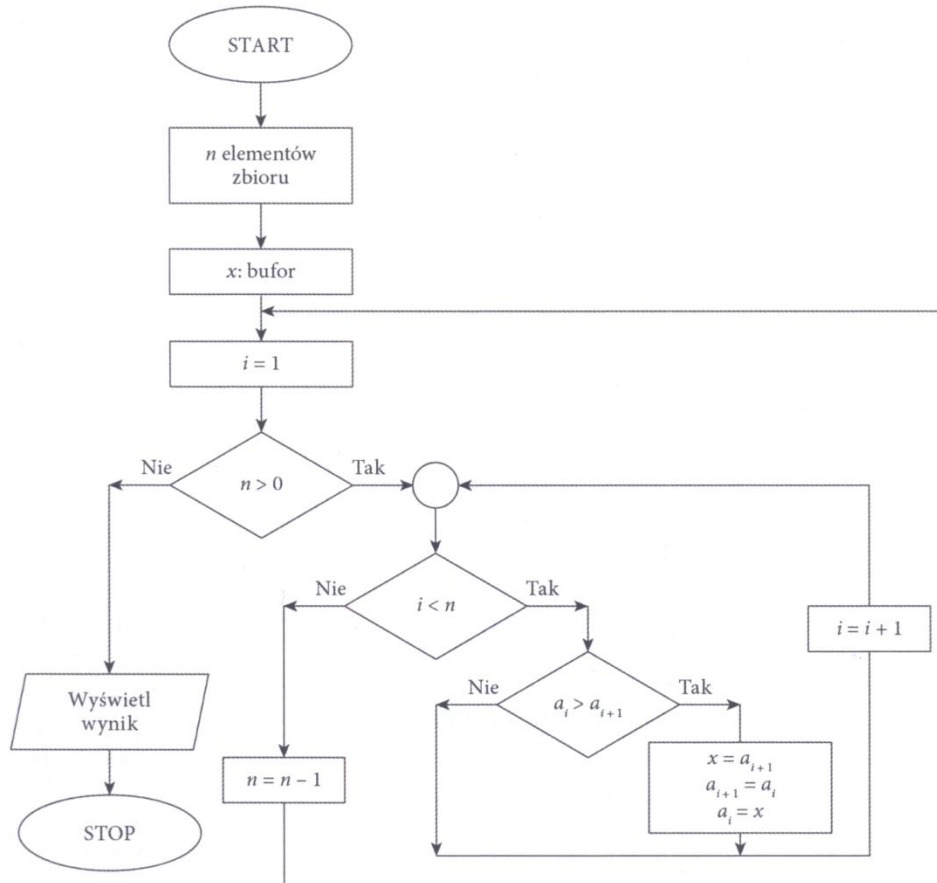


Rysunek 1.5. Algorytm w postaci schematu blokowego

ono na porównywaniu dwóch sąsiednich elementów i zamianie ich miejscami, gdy są ustawione w nieprawidłowej kolejności. Sortowanie kończy się, gdy przy kolejnym przejściu nie ma żadnej zmiany kolejności elementów.

Przykład 1.4

Sortowanie bąbelkowe — schemat blokowy (rysunek 1.6):



Rysunek 1.6. Schemat blokowy sortowania bąbelkowego

Znajdowanie najmniejszego lub największego elementu w zbiorze

Sposób działania algorytmu szybkiego wyszukiwania elementu w zbiorze zależy od tego, czy dane zostały uporządkowane, czy zostały zapisane w przypadkowej kolejności. Jeśli dane są nieuporządkowane, należy przejrzeć wszystkie elementy, aby znaleźć ten właściwy.

Przykład 1.5

Znajdowanie największego elementu w zbiorze nieuporządkowanym:

Dane: n -elementowy zbiór liczb naturalnych

Wynik: max — największa liczba znajdująca się w zbiorze

Krok 1. Przyjmij, że pierwszy element w zbiorze jest największy, czyli $max = a_1$.

Krok 2. Dla kolejnych elementów a_i , gdzie $i = 2, 3, \dots, n$, wykonaj krok 3. oraz krok 4.

Krok 3. Sprawdź, czy max jest mniejsze od a_i .

Krok 4. Jeżeli tak, to dla max przyjmij a_i .

Przykład 1.6

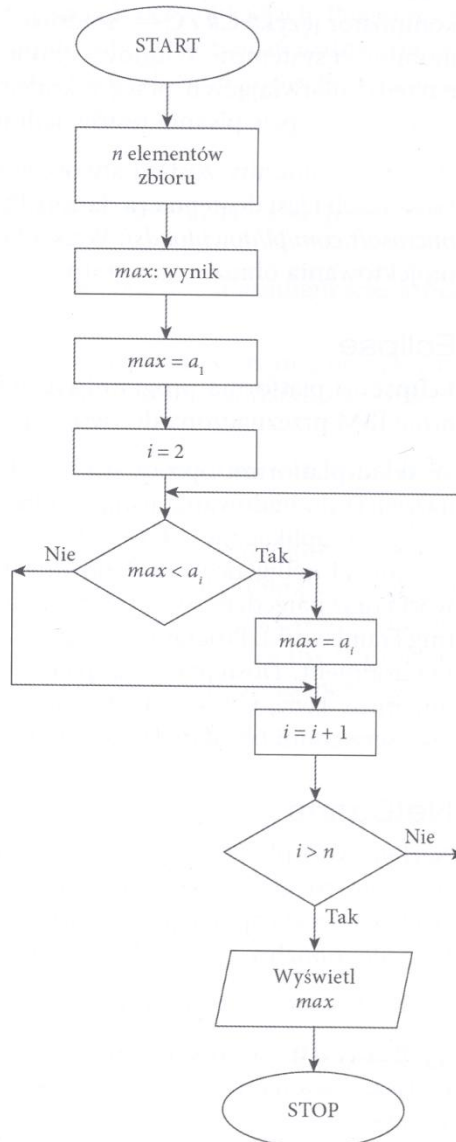
Znajdowanie największego elementu w zbiorze nieuporządkowanym — schemat blokowy (rysunek 1.7).

1.3. Środowiska programistyczne

1.3.1. Zintegrowane środowisko programistyczne (IDE)

Zintegrowane środowisko programistyczne (ang. *Integrated Development Environment*) jest to program (lub zbiór programów) służący do tworzenia, modyfikowania i testowania oprogramowania. Umożliwia tworzenie aplikacji w określonych językach programowania. Najczęściej udostępnia narzędzia obejmujące edycję i kompilowanie kodu źródłowego, narzędzia do tworzenia interfejsu graficznego oraz narzędzia do testowania tworzonego oprogramowania (debugger). Główną zaletą korzystania ze środowiska IDE jest przyspieszenie i usprawnienie pracy nad programem.

Popularnymi środowiskami IDE stosowanymi przy programowaniu są: Microsoft Visual



Rysunek 1.7. Schemat blokowy znajdowania największego elementu w zbiorze nieuporządkowanym

Studio, Code::Blocks, Eclipse, NetBeans, Builder IntelliJ. Pakiety Eclipse i NetBeans to aplikacje, które powstały dla języka Java, ale z czasem dodano do nich wsparcie dla innych języków programowania, na przykład PHP.

Visual Studio

Visual Studio to środowisko programistyczne firmy Microsoft. Zawiera między innymi kompilator języka C# i C++. Środowisko to umożliwia tworzenie oprogramowania dla mobilnych systemów Windows Phone, iOS oraz Android. Oferuje wiele dodatkowych narzędzi ułatwiających pracę z kodem, w tym bardzo rozbudowany debugger, oraz rozszerzone opcje pisania testów jednostkowych.

Wersja Community została stworzona z myślą o studentach i indywidualnych użytkownikach i jest dostępna za darmo. Pakiet można pobrać ze strony <https://visualstudio.microsoft.com/pl/downloads/>. Wersja ta nie zawiera edytora zasobów (brak możliwości projektowania obrazów, ikon itp.).

Eclipse

Eclipse to platforma programistyczna otwartego oprogramowania stworzona przez firmę IBM przeznaczona do tworzenia i testowania aplikacji.

W skład platformy, oprócz środowiska programistycznego, wchodzi między innymi narzędzia do budowania usług i aplikacji sieciowych (Web Tools Platform Project), do rozwijania aplikacji w C/C++ (C/C++ Development Tooling), narzędzie do raportowania (Business Intelligence and Reporting Tools), generator kodu (Eclipse Modeling Framework) oraz narzędzie do tworzenia graficznych interfejsów użytkownika (Graphical Editing Framework). Program wymaga zainstalowanych w systemie bibliotek Java Runtime Environment. Dostępne są implementacje różnych języków programowania, między innymi w Javie, C/C++ i PHP. Oferuje narzędzia do współpracy z serwerami WWW oraz serwerami baz danych. Może pracować z systemami Windows, Linux i macOS.

NetBeans

NetBeans to platforma otwartego oprogramowania, która dostarcza wiele narzędzi programistycznych służących do tworzenia aplikacji, w tym również usług sieciowych (Web Services) i aplikacji na urządzenia mobilne. Przeznaczona jest głównie dla języka Java, ale umożliwia kompilowanie programów napisanych w innych językach.

W skład platformy wchodzi między innymi NetBeans IDE oraz NetBeans Platform.

NetBeans IDE to zintegrowane środowisko programistyczne (IDE) stworzone dla języka Java. Pozwala na tworzenie aplikacji w języku Java oraz aplikacji mobilnych i usług sieciowych.

NetBeans Platform to platforma, która dostarcza gotowe narzędzia (bazę danych, okna, menu, zarządzanie i przechowywanie konfiguracji) umożliwiające tworzenie aplikacji.

1.3.2. Framework — platforma programistyczna

Do tworzenia aplikacji internetowych często wykorzystuje się platformy programistyczne, które definiują strukturę aplikacji, określają mechanizm ich działania oraz dostarczają biblioteki umożliwiające wykonywanie określonych zadań.

Framework ma zdefiniowaną podstawową strukturę aplikacji, czyli elementy, które pozostają niezmiennie we wszystkich utworzonych za jego pomocą aplikacjach. Programista tworzy aplikację, dostosowując poszczególne komponenty do wymagań realizowanego projektu. Ma on również możliwość przygotowania nowych elementów niezbędnych w projektowanej strukturze aplikacji.

Typowe cechy frameworka to:

- *Odwrócenie sterowania* — przepływ sterowania jest narzucany przez framework, a nie przez programistę.
- *Domyślne zachowanie* — framework powinien mieć domyślną konfigurację, która musi być użyteczna i dawać określony wynik.
- *Rozszerzalność* — programista powinien mieć możliwość rozszerzania poszczególnych komponentów frameworka, jeśli chce je rozbudować o dodatkową funkcjonalność.
- *Zamknięta struktura wewnętrzna* — programista może rozbudowywać framework, ale nie poprzez modyfikację domyślnego kodu.

Tworzenie aplikacji z wykorzystaniem frameworków wymaga od programisty napisania mniejszej ilości kodu. Aplikacja zbudowana w ten sposób ma dobrą wewnętrzną strukturę, jest właściwie zaprojektowana i przetestowana. Ceną za elastyczną budowę jest niższa wydajność tworzonego oprogramowania. Frameworki mogą być stosowane jako szkielety zarówno kompletnych aplikacji, jak i pojedynczych komponentów.

Do popularnych frameworków należą:

- Angular,
- React,
- .NET core,
- Django,
- Zend Framework.

Angular

Angular to wspierany i rozwijany przez firmę Google otwarty framework przeznaczony do tworzenia aplikacji jednostronicowych w języku JavaScript. Jego zadaniem jest wdrażanie do aplikacji internetowych wzorca projektowego MVC (ang. *Model-View-Controller*). Zawiera własny kompilator HTML, wbudowany system szablonów i wiele funkcji ułatwiających tworzenie strony (animacje, filtrowanie, elementy interfejsu użytkownika).

React

React to otwarty framework przeznaczony do tworzenia interfejsów graficznych aplikacji jednostronicowych i mobilnych w języku JavaScript. Interfejs użytkownika został oparty na komponentach. Każdy komponent jest funkcją JavaScript zdefiniowaną przez użytkownika. React został wyposażony w język JSX, który jest nakładką na JavaScript i rozszerza jego funkcjonalność.

.NET core

.NET core to wieloplatformowy i otwarty framework przeznaczony do tworzenia aplikacji internetowych z użyciem wzorca MVC. Aplikacje mogą być tworzone za pomocą języka C#, F#, Visual Basic. Framework może działać w systemach Windows, Linux i macOS oraz na urządzeniach mobilnych korzystających z Xamarin. Może być używany do tworzenia aplikacji opartych na chmurze. Taka aplikacja działa w oparciu o mikroserwisy (podprogramy), a jej architektura jest zorientowana na usługi.

Django

Django to otwarty framework przeznaczony do tworzenia aplikacji internetowych napisany w języku Python. Umożliwia przygotowanie aplikacji w oparciu o wzorzec MTV (ang. *Model-Template-View*) pokrewny wzorcowi MVC. Ma prosty system szablonów, automatycznie tworzony panel administracyjny oraz własny serwer do testowania aplikacji.

Zend Framework

Zend Framework to platforma programistyczna przeznaczona do tworzenia aplikacji internetowych w języku PHP. Zawiera zbiór bibliotek obsługujących podstawowe mechanizmy działania aplikacji oraz biblioteki użytkowe obsługujące na przykład wysyłanie e-maili, komunikację z innymi aplikacjami internetowymi itp. Oferuje również implementację modelu MVC, który może zostać wykorzystany do utworzenia struktury aplikacji.



1.4. Pytania i zadania

1.4.1. Pytania

1. Co to jest paradygmat programowania?
2. Wymień zasady programowania strukturalnego.
3. Co to jest typ prosty?
4. Wymień podstawowe cechy języków obiektowych.
5. Na czym polega *polimorfizm*?
6. Wymień podstawowe cechy języków skryptowych.

7. Co to jest *algorytm*?
8. Wymień podstawowe cechy algorytmów.
9. Wymień i omów narzędzia wspomagające tworzenie programów.

1.4.2. Zadania

Zadanie 1.

Utwórz algorytm w postaci listy kroków, schematu blokowego i drzewa algorytmu dla równania liniowego $a \times x + b = 0$.

Zadanie 2.

Utwórz algorytm wyszukiwania najmniejszego elementu w nieuporządkowanym zbiorze danych.