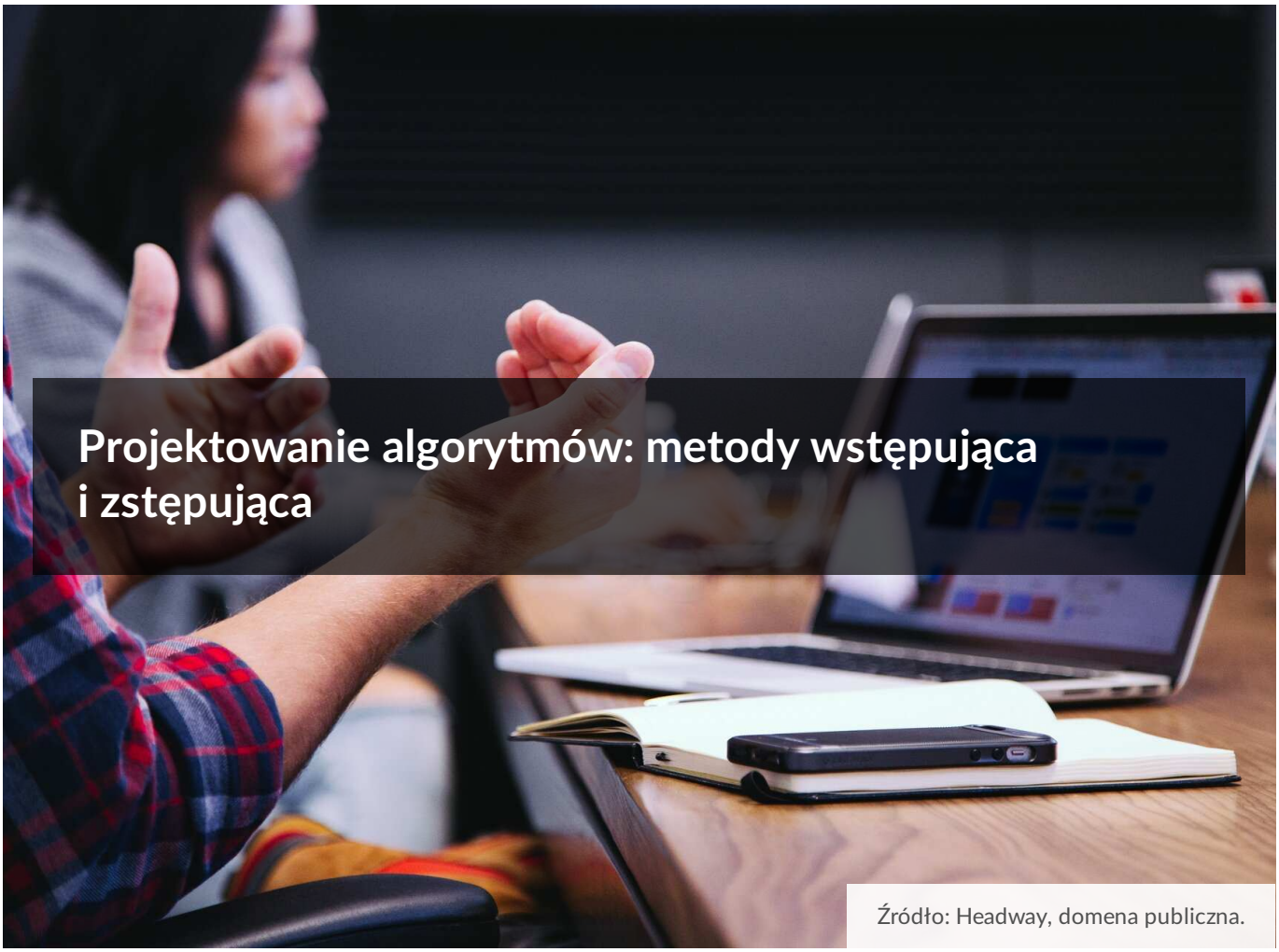
Projektowanie algorytmów: metody wstępująca i zstępująca



Projektowanie algorytmów: metody wstępująca i zstępująca

Źródło: Headway, domena publiczna.

Kiedy spotykamy się z problemami informatycznymi, zdarza się, że nie od razu wiemy, jak je rozwiązać. W takich sytuacjach możemy podzielić zagadnienie na mniejsze podproblemy i dzięki temu zrozumieć, z czym tak naprawdę się mierzymy. Istnieją dwa uniwersalne podejścia do projektowania rozwiązań: są nimi metody **wstępująca** oraz **zstępująca**.

Więcej wskazówek dotyczących rozwiązywania problemów informatycznych znajdziesz w e-materiałach:

- Rozwiązywanie problemów informatycznych – strategia „dziel i zwyciężaj”,
- Jak myśleć komputacyjnie?,
- Rozwiąż to inaczej, czyli myślenie komputacyjne i jego metody.

Twoje cele

- Prześledzisz koncepcje metody wstępującej i metody zstępującej.
- Przeanalizujesz przykładowe zastosowania obu metod.
- Porównasz wady i zalety metody wstępującej oraz zstępującej.

Przeczytaj

Metoda zstępująca

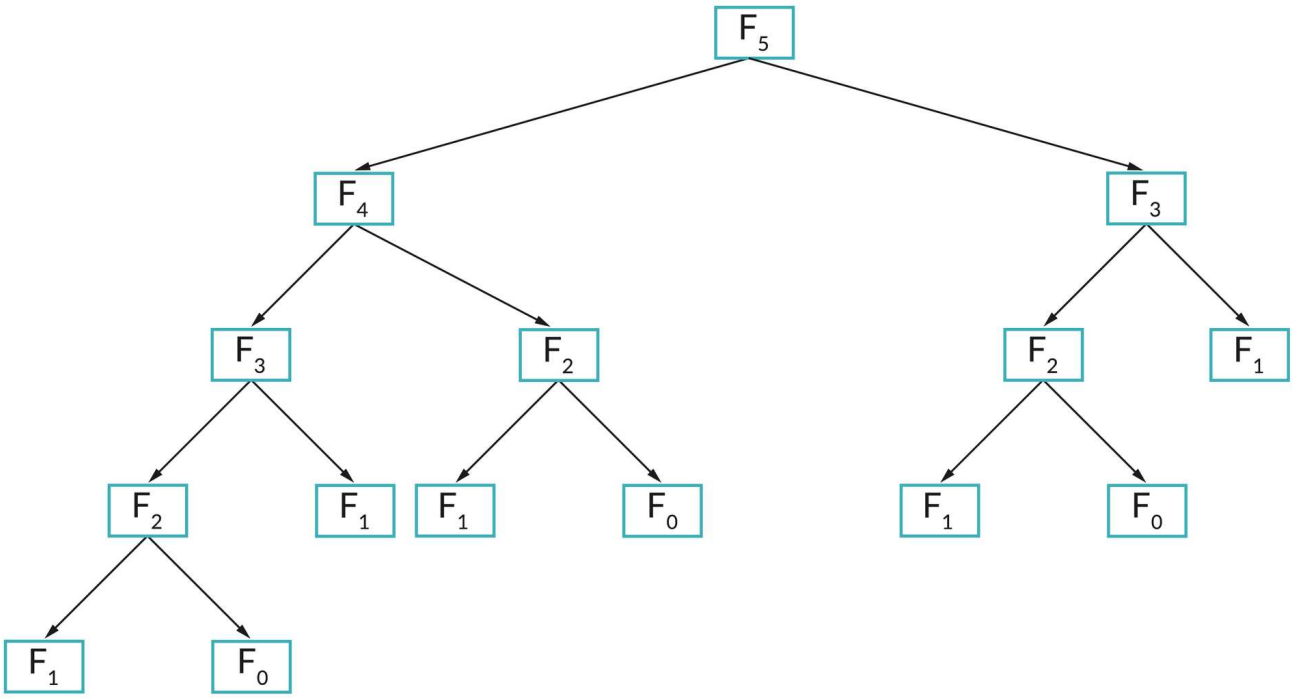
Projektowanie algorytmów – zgodne z metodą zstępującą – polega na rozbijaniu głównego problemu na mniejsze podproblemy. Zanim zaczniemy pisać kod, powinniśmy dokładnie wiedzieć, jakich funkcjonalności potrzebujemy i w jaki sposób mają one ze sobą współdziałać.

Dla lepszego zrozumienia tej koncepcji przeanalizujmy przykład rekurencyjnej metody obliczania wartości elementu ciągu Fibonacciego o indeksie n (więcej informacji o tym zagadnieniu znajdziesz w e-materiale Ciąg Fibonacciego).

> **Ważne!**
>
> Rekurencja to technika programowania polegająca na odwoływaniu się procedur lub funkcji do samych siebie, aż do momentu spełnienia warunku podstawowego. Więcej na jej temat dowiesz się z e-materiału Rekurencja.

By obliczyć wartość elementu o indeksie n, najpierw ustalamy wartość elementu o indeksie $n - 1$ itd.

Na ilustracji zaprezentowano przykład obliczania elementu ciągu Fibonacciego o indeksie 5.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Rozbijając każdy z kroków w taki sposób, przechodzimy do coraz bardziej podstawowych czynności. Postępujemy tak, dopóki nie dojdziemy do bardzo elementarnych operacji,

których już nie jesteśmy w stanie podzielić.

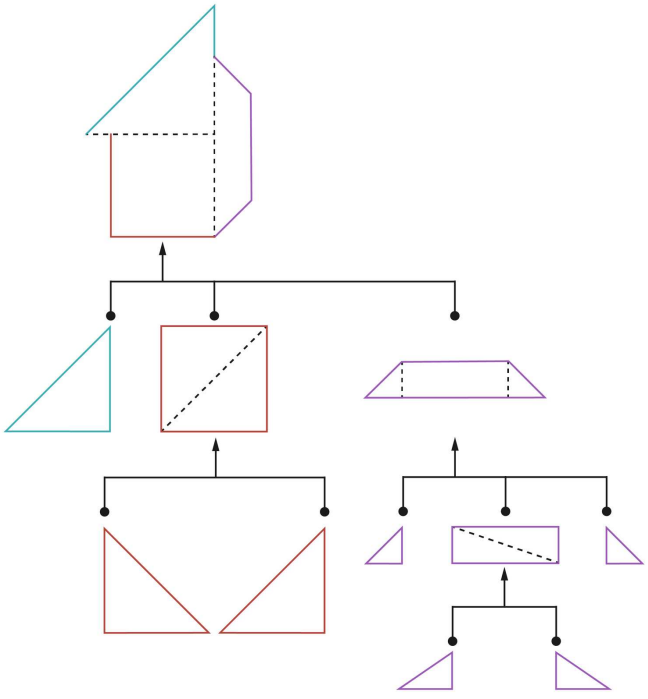
Ten sposób projektowania algorytmów jest dosyć czasochłonny, ponieważ każdy krok powinien być dokładnie zaplanowany i przeanalizowany. Błąd w którymkolwiek kroku powoduje wystąpienie dalszych błędów na niższych poziomach. Należy również przemyśleć algorytm pod względem liczby rozwiązywanych podproblemów. Jeżeli będziemy wykonywać zbyt wiele operacji, złożoność algorytmu znacznie wzrośnie. Plusem tego sposobu jest intuicyjność kolejnych kroków algorytmu.

Metoda wstępująca

Jak sugeruje nazwa, tym razem będziemy tworzyć rozwiązanie, zaczynając od podstawowych funkcjonalności. Metoda ta polega na łączeniu oraz budowaniu zależności pomiędzy elementarnymi funkcjami. Nadajemy w ten sposób kształt coraz to bardziej skomplikowanym i zaawansowanym strukturom, aż dojdziemy do kompletnego programu rozwiązującego problem.

W przypadku obliczenia elementu ciągu Fibonacciego o indeksie `5` zaczynamy od obliczenia pierwszego elementu ciągu Fibonacciego – czyli elementu o indeksie `0` (`n = 0`). Następnie wykorzystujemy znajomość elementu o indeksie `0` do obliczenia elementu o indeksie `1` (`n + 1`) itd.

Inną analogią dla tej metody jest stawianie budowli z klocków. Do klocków możemy porównać elementarne i niezależne od siebie funkcje. Nie powstały one z myślą o konkretnych projektach, dzięki czemu są uniwersalne. Mając je do dyspozycji, jesteśmy w stanie zbudować różnorodne konstrukcje, dowolnie łącząc elementy w całość.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Przedstawiony schemat pokazuje, jak działa metoda wstępująca. Na najniższym poziomie mamy najbardziej podstawowe elementy, które łączymy w coraz bardziej skomplikowane systemy. Na najwyższym poziomie znajduje się cały gotowy system, złożony z elementów, które mieliśmy do dyspozycji na początku.

Paradygmatem programowania korzystającym z metody wstępującej jest **programowanie obiektowe**.

Wady i zalety

W przypadku metody zstępującej jej największą wadą jest czas potrzebny do rozbicia problemu na mniejsze podproblemy. Kolejnym minusem jest brak możliwości przetestowania napisanego programu, dopóki nie powstanie jego większa część. Natomiast bardzo dużym plusem jest dokładne rozpatrzenie problemu i nabycie pewności, że wszystko, co implementujemy, ma cel oraz znajduje zastosowanie. Kolejnym plusem jest to, że niektóre podproblemy mogą okazać się wielokrotnego użytku – ich rozwiązań możemy użyć w więcej niż jednym miejscu, co pozwala zaoszczędzić czas.

Metoda wstępująca pozwala na wygodną przebudowę istniejących rozwiązań. Dodawanie nowych funkcji do istniejącego programu jest proste i nie wymaga edytowania całego kodu, co często ma miejsce w przypadku metody zstępującej. Kolejną zaletą jest łatwe testowanie poszczególnych obiektów w pisanym kodzie, ponieważ z założenia są one od siebie niezależne. Wadą takiego podejścia jest, że z początku może być ono bardzo chaotyczne. Bez dogłębnej analizy problemu nie jesteśmy w stanie stwierdzić, czego tak naprawdę potrzebujemy. Bardzo często zaczynamy zdawać sobie z tego sprawę wraz z pisaniem kodu.

W praktyce metody zstępującej używamy w językach strukturalnych takich jak np. C czy Fortran, natomiast metody wstępującej w językach obiektowych, np. Java czy C++.

> **Ciekawostka**
>
> Zarówno metoda wstępująca, jak i zstępująca znalazły swoje zastosowanie także poza branżą informatyczną. Wykorzystywane są one m.in. w zarządzaniu przy planowaniu struktury podziału pracy (WBS – *Work Breakdown Structure*), w nanotechnologii czy też w neurobiologii.

Słownik

> **paradygmat programowania**
>
> wzorzec implementacyjny definiujący sposób, w jaki programista postrzega przepływ sterowania i wykonywanie programu

Audiobook

Polecenie 1

Po odsłuchaniuaudiobooka wykonaj polecenia.

Audiobook można wysłuchać pod adresem: https://zpe.gov.pl/b/P7AC4SeIf

---

W praktyce możemy wyróżnić dwie metody projektowania (czyli rozwiązywania problemów informatycznych), a w konsekwencji dwie metody programistyczne. Są to:

1. projektowanie zstępujące (top-down design),

2. projektowanie wstępujące (bottom-up design).

Niezależnie od wyboru metody projektowej, dobry projekt powinno przygotowywać się w oparciu o ogólny schemat. Oto najważniejsze zasady, jakimi należy się kierować:

1. zdefiniowanie problemu i określenie wymagań użytkownika;

2. modelowanie projektu, na przykład określenie komponentów, wyróżnienie najistotniejszych z nich oraz podanie specyfikacji wymagań związanych z wydajnością;

3. implementacja;

4. integracja i testowanie systemu;

5. instalacja i testowanie przez użytkowników;

6. utrzymanie, czyli konserwacja.

Projektowanie zstępujące (top-down design)

Projektowanie zstępujące polega na definiowaniu łatwiejszych do rozwiązania podproblemów. „Dekompozycja” ta schodzi niżej, aż do sytuacji, gdy mamy do rozwiązania problem elementarny.

Algorytm postępowania w przypadku projektowania zstępującego prezentuje się następująco:

Pierwszy etap to zdefiniowanie problemu.

Problem dzielimy na dwa główne podproblemy.

Każdy z podproblemów dzielimy na mniejsze podproblemy.

Stopniowo uszczegóławiamy podproblemy, czyli kontynuujemy podział tak długo, aż rozwiązania drobnych podproblemów stają się łatwe.

Następnie dokonujemy złożenia rozwiązań podproblemów niższego rzędu i rozwiązań problemów wyższego rzędu aż do całkowitego rozwiązania problemu.

Najważniejsze zalety projektowania zstępującego to systematyzacja, modularność, przejrzystość kodu oraz możliwość wykorzystania opracowanych rozwiązań do innych programów.

Projektowanie wstępujące (bottom-up design)

Projektowanie wstępujące polega na wyjściu od samego języka. Następnie wzbogaca się go nowymi operacjami do momentu, w którym nie będzie możliwości wyrażenia rozwiązania w rozszerzonym języku. A zatem pisząc program, budujemy „własny język programowania". Spójrzmy na język Lisp. Tworząc program, możemy nie tylko projektować go „w dół" w kierunku języka, ale również tworzyć język w kierunku programu. Przykładowo można stworzyć pewien nowy operator, który w danym miejscu będzie potrzebny. Z czasem może okazać się, że ten nowy operator uprości projekt innej części programu. W takim przypadku, język i program ewoluują razem. W efekcie po skończonym kodowaniu, program będzie wyglądał tak, jakby język programowania był zaprojektowany na jego potrzeby.

Inne wyjaśnienie tej metody odwołuje się do tak zwanych maszyn wirtualnych. Każdy program, który działa na konkretnej maszynie docelowej rozszerza jej możliwości o nowe funkcje. Gdy uruchamiamy program, tak naprawdę tworzymy nową maszynę, która nie wykonuje swoich operacji bezpośrednio, ale zmienia je na prostsze wykonywalne przez sprzęt. Tę nową maszynę nazywamy maszyną wirtualną, ponieważ nie istnieje ona w środowisku fizycznym, tylko abstrakcyjnym.

Analizując omawiane metody rozwiązywania problemów informatycznych, można zauważyć, iż zazwyczaj programy projektuje się łącząc obie te metody. Przystępując bowiem do rozwiązania problemu, w pierwszym kroku rozpoczynamy od podzielenia zadania na podzadania (tu widzimy metodę zstępującą). Następnie dochodzimy do wniosku, że wygodne byłoby zdefiniowanie pakietu funkcji zawierającego potrzebne do wykonania zadania metody. Decydujemy o kształcie tego pakietu i funkcjach wchodzących w jego skład (tu dostrzegamy metodę wstępującą). Powtarzamy to iteracyjnie, za każdym razem dzieląc problemy na prostsze (metoda zstępująca) i rozszerzając język (metoda wstępująca). Powtarzamy schemat do momentu, w którym rozwiązania wszystkich problemów składowych dają się zapisać bezpośrednio w rozszerzonym języku.

> Co ciekawe, metody zstępująca i wstępująca znalazły swoje zastosowanie nie tylko w algorytmice, ale także na przykład w analizie składniowej. Analiza składniowa stanowi jeden z ważniejszych etapów pracy kompilatorów. To w niej sprawdza się kolejność ułożonych leksemów (czyli elementarnych jednostek języka programowania). Zdanie można wyprowadzać od korzenia drzewa to jest wyprowadzenia (czyli aksjomat gramatyki), aż do ostatecznych symboli zdania (analiza zstępująca), bądź rozpoczynając od zdania dojść do aksjomatu gramatyki (metoda wstępująca).

Ważne!

W tym miejscu konieczne jest wyjaśnienie pojęcia modularności. Jego podstawą jest moduł. Moduł czyli pakiet (z angielskiego *module*, *unit*, *package*), to oddzielny twór (najczęściej plik), w którym zapisano interfejs, implementację typów wartości, klasy, zmienne, stałe, funkcje i procedury.

Natomiast modularność to paradygmat programowania, pozwalający na podział kodu na odrębne funkcjonalne części umieszczane w osobnych modułach. Moduły te są niezależne i wymienne.

Modularność wymaga, aby w projektowaniu zstępującym dążyło się do sytuacji, w której podproblemy mogą być rozwiązywane niezależnie, a następnie ich rozwiązania mogły być łączone.

Gdybyśmy chcieli odnaleźć analogię do programowania, powiedzielibyśmy, że modularność wymaga, by poszczególne kroki implementacji realizowane były niezależnie, przy uwzględnieniu, że dane wyjściowe jednego kroku będą wykorzystywane jako dane wejściowe drugiego.

Metoda zstępująca

Przykład 1

W oparciu o technikę zstępującą konstruowane są wszelkiego typu filtry na stronach sklepów internetowych, czy portali aukcyjnych. Przykładem może być system wyszukiwania firm. Jeśli klient chciałby wyszukać sklep w swojej okolicy, nie znając dokładnego adresu, nie powinniśmy wyświetlać mu w systemie komunikatu: podaj adres sklepu. Przeciwnie, podzielimy problem na mniejsze podproblemy i skonstruujemy formularz w oparciu o następujące kroki:

1. wybierz województwo;
2. wybierz miasto;
3. wybierz dzielnicę.

Po wypełnieniu każdego z trzech poziomów, lista wyboru będzie dużo krótsza.

Przykład 2

Algorytm szybkiego sortowania służy do porządkowania tablicy elementów. Jest to jeden z podstawowych algorytmów sortowania.

Działanie algorytmu szybkiego sortowania przedstawia się w następujących krokach:

1. z tablicy wybieramy element rozdzielający;
2. tablica jest dzielona na dwa fragmenty: do początkowego przenoszone są wszystkie elementy nie większe od rozdzielającego, a do końcowego wszystkie większe;
3. sortujemy osobno początkową i końcową część tablicy;
4. dzielenie kończy się, gdy kolejny fragment (uzyskany z podziału) zawiera pojedynczy element, który jest już posortowany.

Metoda wstępująca

Przykład 3

Opracowując sposób policzenia wyrażenia $\left(\frac{11}{28} + \frac{4}{7}\right) \times \frac{5}{6} - 2$, możemy wykorzystać metodę wstępującą. Przy tej metodzie konstrukcji algorytmu wykonujemy następujące kroki:

1. wzbogacamy istniejący język przez dodanie funkcji implementujących działania dodawania, odejmowania, mnożenia i dzielenia na ułamkach;
2. wzbogacamy istniejący język przez dodanie funkcji wczytywania i wypisywania ułamków;
3. wzbogacamy język przez dodanie funkcji skracającej ułamki.

Polecenie 2

Opisz własnymi słowami, na czym polega metoda zstępująca projektowania algorytmów.

Polecenie 3

Spróbuj wskazać podstawową różnicę pomiędzy poznanymi metodami projektowania algorytmów.

Polecenie 4

Postaraj się samodzielnie wskazać zalety analizy zstępującej.

Polecenie 5

Spróbuj samodzielnie wymienić zalety analizy wstępującej.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż, na czym głównie polega koncepcja metody wstępującej.

- ◯ Na rozbijaniu skomplikowanych struktur na podstawowe obiekty.
- ◯ Na dodawaniu nowych funkcjonalności do programu.
- ◯ Na pisaniu czytelnego i zrozumiałego kodu.
- ◯ Na łączeniu podstawowych obiektów w coraz to bardziej skomplikowane struktury.

Ćwiczenie 2

Wskaż, na czym głównie polega koncepcja metody zstępującej.

- ◯ Na projektowaniu optymalnych czasowo rozwiązań.
- ◯ Na dostrzeganiu podobieństw i różnic w schematach.
- ◯ Na rozbijaniu problemu na mniejsze podproblemy.
- ◯ Na zmniejszaniu złożoności obliczeniowej.

Ćwiczenie 3

Wskaż poprawną definicję paradygmatu programowania.

- ◯ Jest to wzorzec projektowy, definiujący sposób podziału głównego problemu na podproblemy.
- ◯ Jest to wzorzec implementacyjny, wyznaczający reguły nazewnictwa funkcji oraz zmiennych.
- ◯ Jest to wzorzec implementacyjny, definiujący sposób patrzenia programisty na przepływ sterowania i wykonywanie programu.
- ◯ Jest to wzorzec projektowy, definiujący sposób optymalizacji czasowej rozwiązań.

Ćwiczenie 4

Uzupełnij zdanie.

Programowanie obiektowe opiera się na [ ].

metodzie zstępującej | metodzie wstępującej

Ćwiczenie 5

Zaznacz wszystkie zalety projektowania rozwiązań z wykorzystaniem metody wstępującej.

- ☐ Proste dodawanie nowych funkcji do istniejącego kodu.
- ☐ Łatwe testowanie poszczególnych obiektów w kodzie.
- ☐ Mała złożoność obliczeniowa.
- ☐ Duża liczba kodu źródłowego.
- ☐ Uzyskanie pewności, że wszystko, co implementujemy, ma swój cel oraz zastosowanie.
- ☐ Mało chaotyczny proces implementacji.

Ćwiczenie 6

Zaznacz wszystkie wady projektowania rozwiązań z zastosowaniem metody zstępującej.

- [ ] Chaotyczny proces implementacji.
- [ ] Nie mamy pewności, czy to, co implementujemy, ma swój cel oraz zastosowanie.
- [ ] Brak dogłębnej analizy problemu.
- [ ] Brak możliwości przetestowania napisanego programu, dopóki nie powstanie jego większa część.
- [ ] Utrudniony proces dodawania nowych funkcji.
- [ ] Duża ilość czasu potrzebna na rozbicie problemu na mniejsze podproblemy.

Ćwiczenie 7

Zapoznaj się z algorytmami. Podpisz je odpowiednio, decydując, czy prezentują metodę zstępującą czy wstępującą.

Specyfikacja problemu:

Dane:

- n – liczba naturalna; indeks ciągu Fibonacciego

Wynik:

- wyraz ciągu Fibonacciego o indeksie n

Przykład 1

```
funkcja Fibonacci(n):
    F[0] ← 0
    F[1] ← 1
    dla i = 2, 3, ..., n wykonuj:
        F[i] ← F[i − 1] + F[i − 2]
    zwróć F[n]
```

Przykład 2

```
funkcja F(n):
    jeżeli n > 1 wykonaj:
        zwróć F(n − 1) + F(n − 2)
    zwróć n
```

Ćwiczenie 8

Wstaw każdy element do najlepiej pasującej dla niego grupy.

Metoda wstępująca

Metoda zstępująca

- tworzenie budowli z klocków dla dzieci
- łączenie podstawowych funkcji
- rozbijanie głównego problemu na mniejsze podproblemy
- pewność, że nie zabraknie żadnej krytycznej funkcji
- duża ilość czasu poświęcona na planowanie
- programowanie obiektowe
- budowanie zależności pomiędzy elementarnymi funkcjami

Dla nauczyciela

Autor: Bartosz Zadrożny
Przedmiot: Informatyka

Temat: Projektowanie algorytmów: metody wstępująca i zstępująca

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

3) wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) w zależności od problemu rozwiązuje go, stosując metodę wstępującą lub zstępującą;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Prześledzisz koncepcje metody wstępującej i metody zstępującej.

- Przeanalizujesz przykładowe zastosowania obu metod.
- Porównasz wady i zalety metody wstępującej oraz zstępującej.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Projektowanie algorytmów: metody wstępująca i zstępująca". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Ustalenie celu lekcji i kryteriów sukcesu.
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie przystępują do cichego czytania tekstu zawartego w sekcji „Przeczytaj", jeśli nauczyciel - na podstawie raportu na platformie - uważa, że przygotowanie uczniów jest wystarczające, może pominąć tę czynność.

2. **Praca z multimedium.** Uczniowie wspólnie słuchają audiobooka i starają się odpowiedzieć na pytania:
   – Opisz własnymi słowami, na czym polega metoda zstępująca projektowania algorytmów?
   – Spróbuj wskazać podstawową różnicę pomiędzy poznanymi metodami projektowania algorytmów?
   – Własnymi słowami postaraj się wskazać zalety metody zstępującej.
   – Własnymi słowami postaraj się wskazać zalety metody wstępującej.
3. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1-5 z sekcji „Sprawdź się", po ich wykonaniu porównują otrzymane wyniki z inną osobą.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie wykonują ćwiczenia 6-8 z sekcji „Sprawdź się".

Wskazówki metodyczne:

- Treści w sekcji „Audiobook" można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.