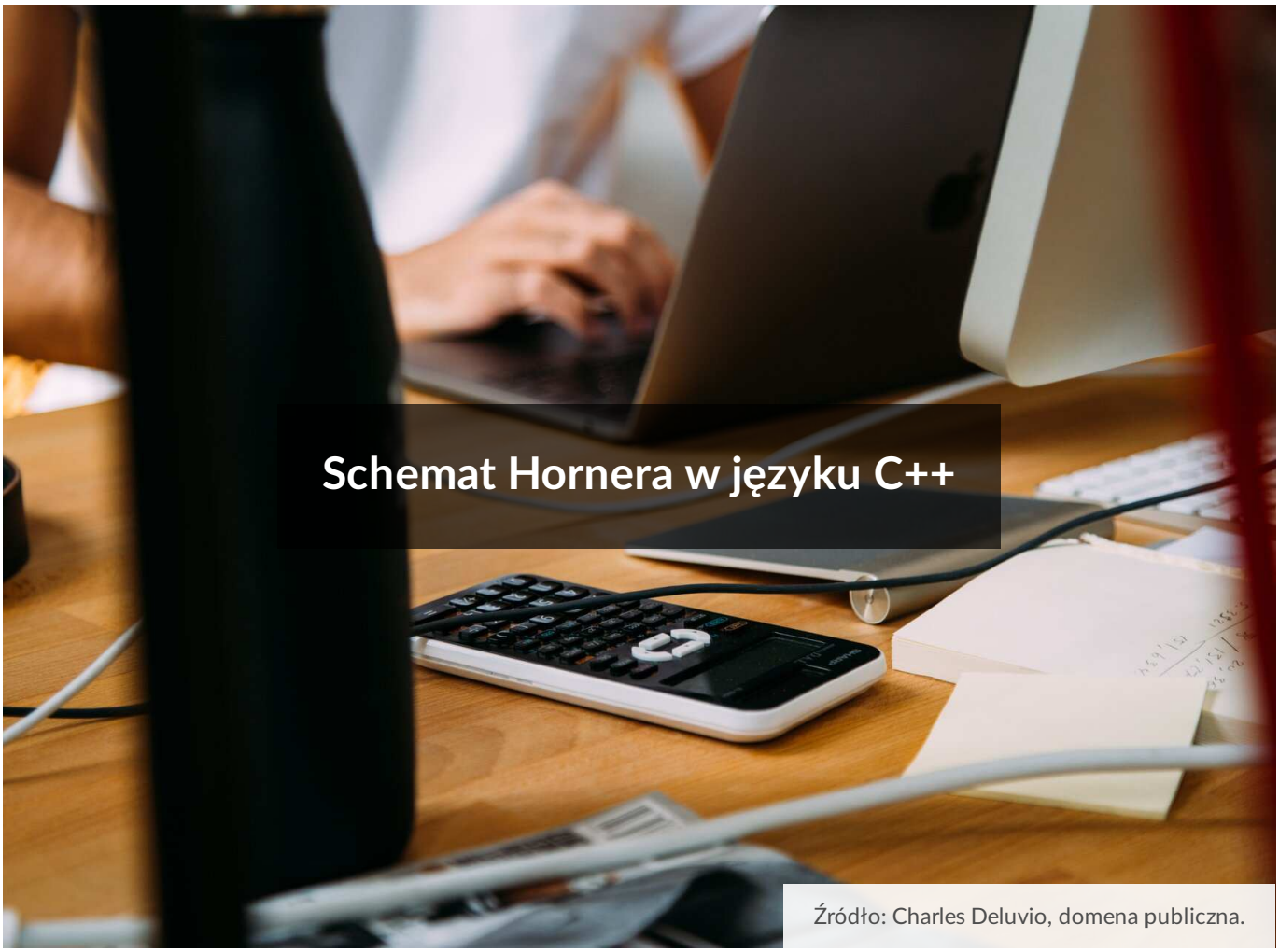
Schemat Hornera w języku C++



Źródło: Charles Deluvio, domena publiczna.

W e-materiale Schemat Hornera poznaliśmy dwie wersje algorytmu, za pomocą którego można obliczyć wartość wielomianu, wykorzystując minimalną liczbę mnożeń.

W tym e-materiale zaimplementujemy schemat Hornera w języku C++.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Schemat Hornera w języku Java.
- Schemat Hornera w języku Python.

Więcej zadań znajdziesz w materiale: Schemat Hornera – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz algorytm realizujący schemat Hornera iteracyjnie i rekurencyjnie w języku C++.
> - Zaimplementujesz algorytm w wersji iteracyjnej oraz rekurencyjnej, obliczający wartość wielomianu za pomocą schematu Hornera.
> - Zastosujesz algorytm wykorzystujący schemat Hornera, by rozwiązać konkretne problemy.

Przeczytaj

Schemat Hornera jest sposobem obliczania wartości wielomianu dla podanego argumentu. Dzięki tej metodzie ograniczamy liczbę mnożeń, którą musimy wykonać w celu rozwiązania problemu. W tym materiale zaimplementujemy w języku C++ dwie wersje algorytmu: iteracyjną oraz rekurencyjną.

Specyfikacja problemu:

Dane:

- `wspolczynniki[]` – tablica zawierająca liczby rzeczywiste (kolejne współczynniki wielomianu)
- `stopienWielomianu` – liczba naturalna
- `x` – liczba rzeczywista; argument, dla którego obliczana jest wartość wielomianu

Wynik:

- `wynik` – liczba rzeczywista

Przykładowe wyjście:

```
Wartość wielomianu dla argumentu x = 4
```

Schemat Hornera – wersja iteracyjna

Zapiszmy krok po kroku algorytm obliczający wartość wielomianu w wersji iteracyjnej.

Krok 1.

Zapiszmy nagłówek naszej funkcji.

Najpierw zapisujemy typ zwracanych wartości przez naszą funkcję. W tym przypadku może to być zarówno typ `int`, `float`, `double`, jak i `void`. W przypadku funkcji typu `void` musimy w ciele funkcji zawrzeć wypisywanie na ekran obliczonej wartości.

Następnie zapisujemy nazwę naszej funkcji `horner`, a potem w okrągłych nawiasach zapisujemy argumenty funkcji. Pierwszym z argumentów jest tablica `wspolczynniki` zawierająca, jak sama nazwa wskazuje, wszystkie współczynniki znajdujące się przy kolejnych potęgach naszego wielomianu. Kolejnym argumentem jest `stopienWielomianu`. Jest to zmienna, w której przechowywana jest największa potęga naszego wielomianu. Ostatnim argumentem jest zmienna `x`. Jest to argument, dla którego będziemy obliczać wartość wielomianu.

```
int hornerIteracyjnie(int wspolczynniki[],int stopienWielomianu,
```

Krok 2.

Deklarujemy zmienną `wynik`. Może być ona zarówno typu `int`, `double`, jak i `float`.

```
int wynik;
```

Krok 3.

Kolejnym krokiem będzie przypisanie do zmiennej `wynik` wartości równej wyrazowi wolnemu w naszym wielomianie.

```
wynik = wspolczynniki[0];
```

Krok 4.

Następnie tworzymy pętlę wykonującą się od `i = 1` do `i = stopienWielomianu`. Zaczynamy pętlę od wartości równej 1, ponieważ wartość współczynnika wyrazu wolnego została uwzględniona w obliczeniach poza pętlą. Pętla wykonuje się do wartości równej `stopienWielomianu`, ponieważ zaczynamy od stopnia równego 1 i wykonujemy pętlę do momentu, w którym obliczymy cały wielomian.

```
for(int i = 1; i <= stopienWielomianu; i++)
```

Krok 5.

W pętli do zmiennej `wynik` przypisujemy jej obecną wartość przemnożoną przez argument `x`, a do tego dodajemy zmienną zapisaną w tabeli `wspolczynniki` pod indeksem `i`.

```
wynik = wynik * x + wspolczynniki[i];
```

Krok 6.

Po wykonaniu pętli dla funkcji typu `void` wypisujemy wartość zmiennej `wynik` w odpowiednim komunikacie.

```
cout<<"Wartość wielomianu dla argumentu x = "<<wynik;
```

Dla funkcji typu całkowitego (`int`) zwracamy zmienną `wynik`

```
return wynik;
```

Cały algorytm, włącznie z krokiem 6. zapisanym dla funkcji typu całkowitego, prezentuje się następująco:

```
int hornerIteracyjnie(int wspolczynniki[],int stopienWielomianu,
{
    int wynik = wspolczynniki[0];

    for(int i = 1; i <= stopienWielomianu; i++)
        wynik = wynik * x + wspolczynniki[i];

    return wynik;
}
```

Sprawdźmy jego działanie dla przykładowych danych. Przetestujemy działanie programu dla następującego wielomianu:

$W(x) = x^2 + 3x - 6$

oraz $x = 2$.

```
#include<iostream>
using namespace std;

int hornerIteracyjnie(int wspolczynniki[],int stopienWielomianu,
{
    int wynik = wspolczynniki[0];

    for(int i = 1; i <= stopienWielomianu; i++)
        wynik = wynik * x + wspolczynniki[i];

    return wynik;
}

int main()
{
    int wspolczynniki[] = {1, 3, -6};
```

```
    int stopienWielomianu = 2;
    int x = 2;
    cout << hornerIteracyjnie(wspolczynniki, stopienWielomianu, x
    return 0;
}
```

Program na wyjściu daje liczbę 4, co jest poprawnym wynikiem.

Schemat Hornera – wersja rekurencyjna

Zapiszmy, krok po kroku, algorytm schematu Hornera w wersji rekurencyjnej.

Krok 1.

Zapiszmy nagłówek naszej funkcji:

```
int hornerRekurencyjnie(int wspolczynniki[],int stopienWielomianu
```

Składa się on z następujących elementów:

- typu zwracanych wartości przez funkcję – w tym przypadku może to być zarówno `int`, `float`, jak i `double`,
- nazwy funkcji `hornerRekurencyjnie`,
- argumentów funkcji zapisanych w okrągłych nawiasach.

Pierwszym z argumentów jest tablica `wspolczynniki` zawierająca wszystkie współczynniki znajdujące się przy kolejnych potęgach naszego wielomianu. Kolejnym argumentem jest `stopienWielomianu`. Jest to zmienna, w której przechowywana jest największa potęga naszego wielomianu. Ostatnim argumentem jest zmienna `x`. Jest to argument, dla którego będziemy obliczali wartość wielomianu.

Krok 2.

Kolejnym krokiem jest ustalenie warunku przerwania rekurencji. Tym warunkiem będzie to, czy zmienna `stopienWielomianu` jest równa 0.

```
if(stopienWielomianu == 0)
```

Krok 3.

Jeśli warunek zapisany w **kroku 2** jest spełniony, zwracamy współczynnik z indeksem zero – jest to wyraz wolny naszego wielomianu.

```
return wspolczynniki[0];
```

Krok 4.

Jeśli nasz warunek nie jest spełniony, zwracamy argument, dla którego obliczamy wartość wielomianu pomnożoną przez wartość, jaką zwróci nam wywołana rekurencyjnie funkcja `hornerRekurencyjnie()` z argumentami `wspolczynniki`, `stopienWielomianu - 1`, `x` oraz dodanym współczynnikiem zapisanym pod indeksem `stopienWielomianu`.

```
return x * hornerRekurencyjnie(wspolczynniki,stopienWielomianu -
```

Algorytm obliczający wartość wielomianu zapisany w wersji rekurencyjnej zaimplementowany w języku C++ wygląda następująco:

```
int hornerRekurencyjnie(int wspolczynniki[],int stopienWielomianu
{

    if(stopienWielomianu == 0)
        return wspolczynniki[0];
    else
        return x * hornerRekurencyjnie(wspolczynniki,stopienWielo
}
```

Sprawdźmy jego działanie dla przykładowych danych. Przetestujemy działanie programu dla następującego wielomianu:

$W(x) = x^2 + 3x - 6$

oraz $x = 2$.

```
#include<iostream>
using namespace std;

int hornerRekurencyjnie(int wspolczynniki[],int stopienWielomianu
{
```

```
6
7       if(stopienWielomianu == 0)
8           return wspolczynniki[0];
9       else
10          return x * hornerRekurencyjnie(wspolczynniki,stopienWielo
11  }
12
13  int main()
14  {
15      int wspolczynniki[] = {1, 3, -6};
16      int stopienWielomianu = 2;
17      int x = 2;
18      cout << hornerRekurencyjnie(wspolczynniki, stopienWielomianu,
19
20      return 0;
21  }
```

Program na wyjściu daje liczbę 4, co jest poprawnym wynikiem.

Słownik

> **algorytm**
>
> przepis postępowania prowadzący do rozwiązania ustalonego problemu, określający ciąg czynności elementarnych, które należy w tym celu wykonać

> **iteracja**
>
> technika programowania polegająca na powtarzaniu tych samych operacji w pętli określoną liczbę razy lub do momentu, aż zostanie spełniony zadany warunek

> **przypadek podstawowy**
>
> przypadek, dla którego funkcja rekurencyjna nie wywołuje samej siebie, a zamiast tego zwraca z góry określoną wartość

> **rekurencja**
>
> technika programowania polegająca na tym, że funkcja wywołuje samą siebie, aż do napotkania przypadku podstawowego

> **stopień wielomianu**
>
> najwyższa potęga przy zmiennej, która nie jest równa zero

wielomian

wyrażenie algebraiczne będące sumą jednomianów, czyli wyrażeń postaci $a_n x^n$

William George Horner

brytyjski matematyk żyjący w latach 1786–1837; w wieku 14 lat został asystentem nauczyciela w szkole w Bristolu, a w 1809 roku założył własną szkołę w Bath; podał sposób wyznaczania wartości wielomianu z wykorzystaniem minimalnej liczby operacji możenia

wykonanie elementarne w rekurencji

moment, gdy funkcja rekurencyjna zwraca konkretną wartość zamiast kolejnego wywołania rekurencyjnego

Prezentacja multimedialna

Ćwiczenie 1

Zapisz algorytm dzielenia wielomianu przez dwumian z wykorzystaniem schematu Hornera, wykorzystując pseudokod lub język programowania.

Przetestuj jego działanie dla następującego dzielenia wielomianu przez dwumian:

$$4x^4 - 3x^2 + x - 7 \ : \ x - 2$$

Wykonanie tej operacji jako wynik da nam tablicę współczynników wielomianu, będącego wynikiem dzielenia zadanego wielomianu przez dwumian oraz resztę z dzielenia. Dla przedstawionego przykładu tablica ta będzie zawierała elementy: 4, 8, 13, 27, 47. Ponieważ wynikiem przedstawionego działania jest wielomian
$4x^3 + 8x^2 + 13x + 27 \ reszta \ z \ dzielenia \ = \ 47.$

Specyfikacja problemu:

Dane:

- `wspolczynniki` – tablica liczb rzeczywistych przechowująca współczynniki wielomianu (dzielnej)
- `stopienWielomianu` – liczba naturalna
- `dwumian` – wyraz wolny dwumianu (dzielnika); liczba rzeczywista

Wynik:

- `wynik` – liczba rzeczywista

Polecenie 1

Przeanalizuj prezentację przedstawiającą dzielenie wielomianu przez dwumian i porównaj z nią swoje rozwiązanie.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Pierwszym krokiem, jaki musimy wykonać, jest zapisanie nagłówka funkcji:

```
void horner(double wspolczynniki[], int stopienWielomianu, double dwumian)
```

Przeanalizujmy powyższy nagłówek element po elemencie:

`void` – rodzaj funkcji, która nie zwraca nic,

`horner` – nazwa funkcji,

`double` – typ zmiennej liczbowej przechowującej liczby zmiennoprzecinkowe,

`wspolczynniki[]` – tablica przechowująca współczynniki wielomianu, który jest dzielną,

`int` – typ zmiennej liczbowej przechowującej liczby całkowite,

`stopienWielomianu` – zmienna przechowująca stopień wielomianu, który jest dzielną,

`dwumian` – zmienna przechowująca wyraz wolny dwumianu, który jest dzielnikiem.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Kolejnym krokiem będzie zadeklarowanie tablicy, w której będziemy przechowywać wynik dzielenia.

```
double * wynik = new double
[stopienWielomianu + 1];
```

Tablica `wynik` ma `stopienWielomianu + 1` komórek, ponieważ w pierwszych `stopienWielomianu` komórkach będzie zapisany wynik naszego dzielenia, zaś w ostatniej komórce będzie zapisana reszta z dzielenia.

Tablicę deklarujemy, wypełniając ją od razu zerami, by nie było w niej danych, które zostały wcześniej w pamięci.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Następnie ustawmy wartość komórki o indeksie zero w tablicy `wynik` na wartość komórki o tym samym indeksie w tablicy `wspolczynniki`.

```
wynik[0] = wspolczynniki[0];
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Kolejnym krokiem będzie stworzenie pętli `for` tak, by przejść przez każdy element obu tablic. Aby to zrobić, musimy zacząć pętlę od `i` równego jeden (ponieważ komórkami o indeksie zero zajęliśmy się wcześniej) do `i <= stopienWielomianu` (ponieważ tablica ma `stopienWielomianu + 1` komórek).

```
for (int i = 1; i <=
stopienWielomianu; i++)
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

W pętli zapisanej w poprzednim kroku musimy obliczyć wartość wielomianu, który powstanie poprzez podzielenie dzielnej przez dzielnik. Robimy to, zapisując równanie: `wynik[i] = dwumian * (-1) * wynik[i - 1] + wspolczynniki[i];`

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Po wykonaniu pętli w tablicy `wynik[]` mamy obliczony iloraz, a w komórce o indeksie `stopienWielomianu` mamy zapisaną resztę z dzielenia.

Wyświetlmy wielomian. Aby to zrobić, najpierw musimy stworzyć pętlę `for`, która pozwoli nam wyświetlić cały wielomian. Będzie się ona wykonywała od `i = 0` dopóki `i < stopienWielomianu`. Pętla ta nie wykonuje się do końca tablicy, ponieważ w ostatniej komórce zawarta jest reszta z dzielenia.

```
1
2 for (int i = 0; i < stopienWielomianu; i++)
3 {
4 }
5
```

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Umieśćmy w pętli komunikat pozwalający nam wyświetlić kolejne współczynniki przy odpowiednich potęgach zmiennej x w naszym wielomianie. Aby to zrobić, zapiszmy:

```
cout << wynik[i] << "x^" <<
stopienWielomianu - i - 1;
```

Pierwsze, co wyświetlamy, to odpowiedni współczynnik obliczony wcześniej. Następnie wyświetlamy oznaczenie x do potęgi. Ostatnim, co wyświetlamy, jest potęga, do której podniesiona jest wartość zmiennej x w naszym wielomianie. Potęgę obliczamy poprzez odjęcie od `stopienWielomianu` zmiennej `i`, ponieważ zmienna `i` zwiększa się o jeden co każdy obrót pętli. Finalnie od całości odejmujemy jeden. Robimy to, ponieważ w zmiennej `stopienWielomianu` zawarty jest stopień wielomianu, który był dzielną, a iloraz jest stopnia o jeden mniejszego, ponieważ dzieliliśmy dzielną przez dwumian.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Następnym krokiem będzie sprawdzenie, czy powinniśmy wyświetlić znak dodawania. Powinien się on wyświetlić w przypadku, gdy następny współczynnik wielomianu jest większy bądź równy zero oraz nie jest on zapisany w komórce o indeksie równym `stopienWielomianu`. Gdyby tak było, oznaczałoby to, że jest to reszta z dzielenia.

```
if (wynik[i + 1] >= 0 && (i + 1)
!= stopienWielomianu)
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Jeśli warunek sprawdzany w poprzednim kroku jest spełniony, wyświetlamy znak dodawania.

```
cout << "+";
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Na końcu, poza pętlą, wyświetlmy wartość reszty z dzielenia. Jest ona zapisana w komórce o indeksie `stopienWielomianu`.

```
cout << " reszta z dzielenia = "
<< wynik[stopienWielomianu] <<
endl;
```

Musimy też pamiętać o usunięciu z pamięci dynamicznie zdefiniowanej tabeli.

```
delete [] wynik;
```

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P12dbUcaL

Cały algorytm wygląda następująco:

```
1 #include <iostream>
2 using namespace std;
3
4 void horner(double
  wspolczynniki[], int
```

```
stopienWielomianu,
double dwumian)
{
    double *wynik = new
double[stopienWielomianu
+ 1];

    wynik[0] =
wspolczynniki[0];
    for (int i = 1; i <=
stopienWielomianu; i++)
    {
        wynik[i] =
dwumian * (-1) * wynik[i
- 1] + wspolczynniki[i];
    }

    for (int i = 0; i <
stopienWielomianu; i++)
    {
        cout << wynik[i]
<< "x^" <<
stopienWielomianu - i -
1;
        if (wynik[i + 1]
>= 0 && (i + 1) !=
stopienWielomianu)
            cout << "+";
    }
    cout << " reszta z
dzielenia = " <<
wynik[stopienWielomianu]
<< endl;
    delete [] wynik;
}

int main()
{
    double
wspolczynniki[] = { 4,
0, -3, 1, -7 };

horner(wspolczynniki, 4,
-2);
    return 1;
```

```
29 }
```

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który obliczy wartość wielomianu, zgodnie ze schematem Hornera (metoda iteracyjna). Przetestuj jego działanie dla następującego wielomianu $5x^3 + 2x^2 + x + 10$ dla argumentu $x = 5$.

Specyfikacja:

Dane:

- `wsp` – tablica liczb całkowitych; współczynniki wielomianu
- `stopien` – liczba całkowita; stopień wielomianu
- `x` – liczba całkowita, argument

Wynik:

- `y` – liczba całkowita; wartość wielomianu

Ćwiczenie 2

Napisz program, który obliczy wartość wielomianu, zgodnie ze schematem Hornera (metoda rekurencyjna). Przetestuj jego działanie dla wielomianu $x^5 + 7x^3 + x + 1$ dla argumentu $x = 10$.

Specyfikacja:

Dane:

- `wsp` – tablica liczb całkowitych; współczynniki wielomianu
- `stopien` – liczba całkowita; stopień wielomianu
- x – liczba całkowita, argument

Wynik:

- y – liczba całkowita; wartość wielomianu

Ćwiczenie 3

Napisz program, który obliczy, a następnie wyświetli kolejne współczynniki wielomianu podzielonego przez dwumian. Przetestuj działanie programu dla wielomianu $6x^3 - x^2 + 5x + 12$ podzielonego przez dwumian $x - 5$.

Specyfikacja:

Dane:

- `wsp` – tablica liczb całkowitych; współczynniki wielomianu
- `stopien` – liczba całkowita; stopień wielomianu
- `dwumian` – liczba całkowita
- `x` – liczba całkowita; argument

Wynik:

- `y` – tablica liczb całkowitych; kolejne współczynniki wielomianu podzielonego przez dwumian

Ćwiczenie 4

Aby obliczyć wartość liczby a podanej w systemie liczbowym o podstawie x, możemy obliczyć wartość wielomianu

$$w(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \ldots + a_{n-1}x^1 + a_nx^0$$

w którym a_i to i-ta cyfra liczby a (gdzie a_0 to cyfra najbardziej znacząca liczby a), a n to liczba cyfr w liczbie a pomniejszona o 1.

Napisz program, który dokona zamiany liczby podanej w systemie liczbowym z przedziału [2, 9] na liczbę w systemie dziesiętnym zaprezentowanym sposobem. W celu obliczenia wartości wielomianu zastosuj algorytm wykorzystujący schemat Hornera.

Przetestuj działanie programu dla liczby w systemie trójkowym o wartości *21012211.*

Specyfikacja:

Dane:

- `liczba` – ciąg znaków; liczba całkowita zapisana w systemie liczbowym z przedziału [2, 9]

Wynik:

- `wynik` – liczba całkowita; liczba w systemie decymalnym

Dla nauczyciela

Autor: Maurycy Gast

Przedmiot: Informatyka

Temat: Schemat Hornera w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:
Zakres podstawowy i rozszerzony
Cele kształcenia – wymagania ogólne
1) Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.
2) Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.
Zakres rozszerzony
II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
3. sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;
III. I + II. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
1. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:
8) obliczania wartości wielomianu za pomocą schematu Hornera,

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):

- Przeanalizujesz algorytm realizujący schemat Hornera iteracyjnie i rekurencyjnie w języku C++.

- Zaimplementujesz algorytm w wersji iteracyjnej oraz rekurencyjnej, obliczający wartość wielomianu za pomocą schematu Hornera.
- Zastosujesz algorytm wykorzystujący schemat Hornera, by rozwiązać konkretne problemy.

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;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Schemat Hornera w języku C++”. Uczniowie zapoznają się z treściami w sekcji „Przeczytaj” w kontekście programowania.

Faza wstępna:

1. Chętna lub wybrana osoba przedstawia najważniejsze informacje związane ze schematem Hornera.
2. Nauczyciel wyświetla temat oraz cele zajęć, omawiając lub ustalając razem z uczniami kryteria sukcesu.

3. **Rozpoznanie wiedzy uczniów.** Uczniowie tworzą pytania dotyczące tematu zajęć, na które odpowiedzą w trakcie lekcji.

Faza realizacyjna:

1. Nauczyciel sprawdza przygotowanie uczniów do zajęć, prosząc chętną lub wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj".
2. **Praca z tekstem.** Nauczyciel prosi uczniów, by w parach przeanalizowali zaproponowane w sekcji „Przeczytaj" rozwiązania i przetestowali ich działanie na podstawie podanych przez siebie wielomianów.
3. **Praca z multimedium**. Uczniowie zapoznają się z Ćwiczeniem 1 z sekcji „Prezentacja multimedialna". W zależności od grupy nauczyciel może zaproponować napisanie opisanego programu za pomocą języka programowania lub pseudokodu. Uczniowie porównują swoje rozwiązania z prezentacją.
4. **Ćwiczenie umiejętności.** Uczniowie, pracując w parach, wykonują ćwiczenie nr 1 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność pisanych kodów, porównuje je i omawia wraz z uczniami. Wskazuje najbardziej efektywne rozwiązanie.
5. Liga zadaniowa – uczniowie pracując w parach wykonują ćwiczenie nr 2 z sekcji „Sprawdź się", a następnie dzielą się swoimi wynikami przez porównywanie napisanego kodu z inną grupą, która również zakończyła zadanie.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.
2. Na koniec zajęć z programowania w C++ nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Uczniowie wykonują ćwiczenia 3–4 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj" można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.