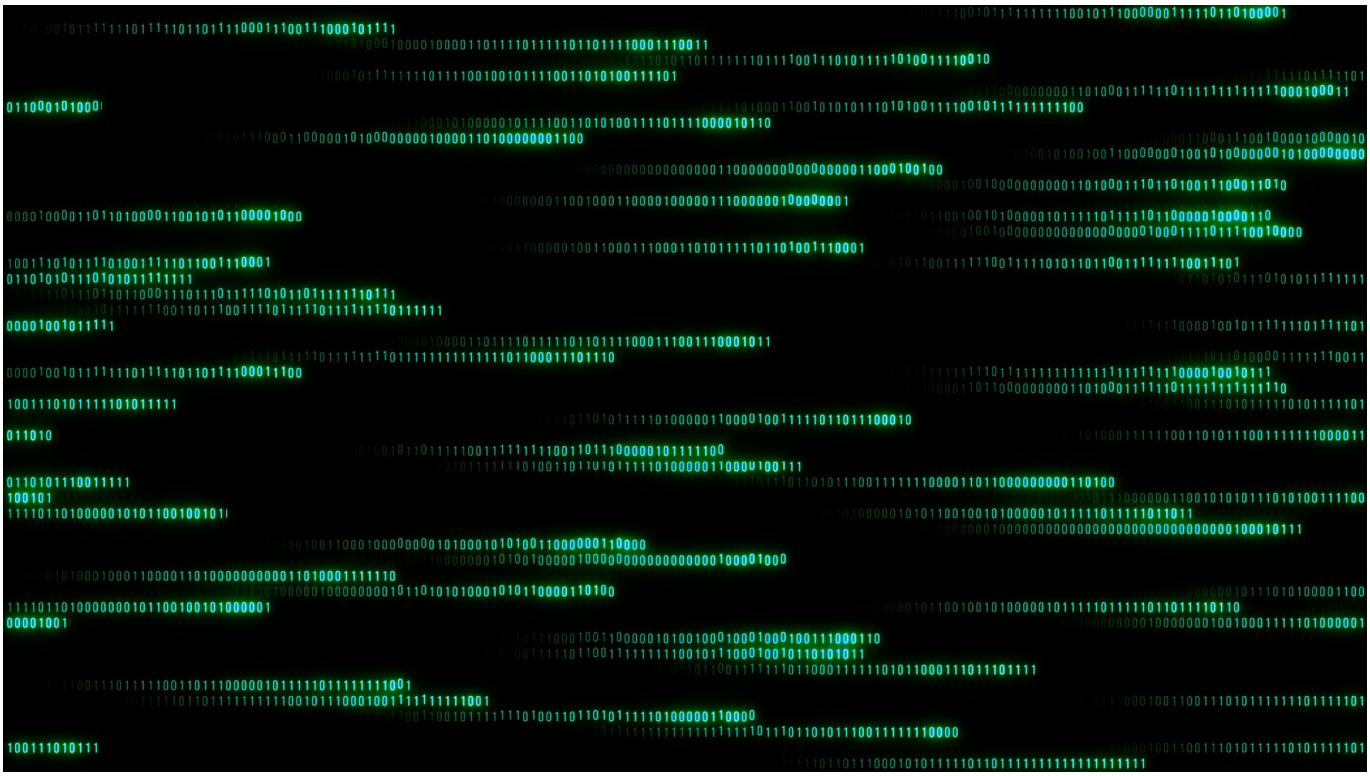
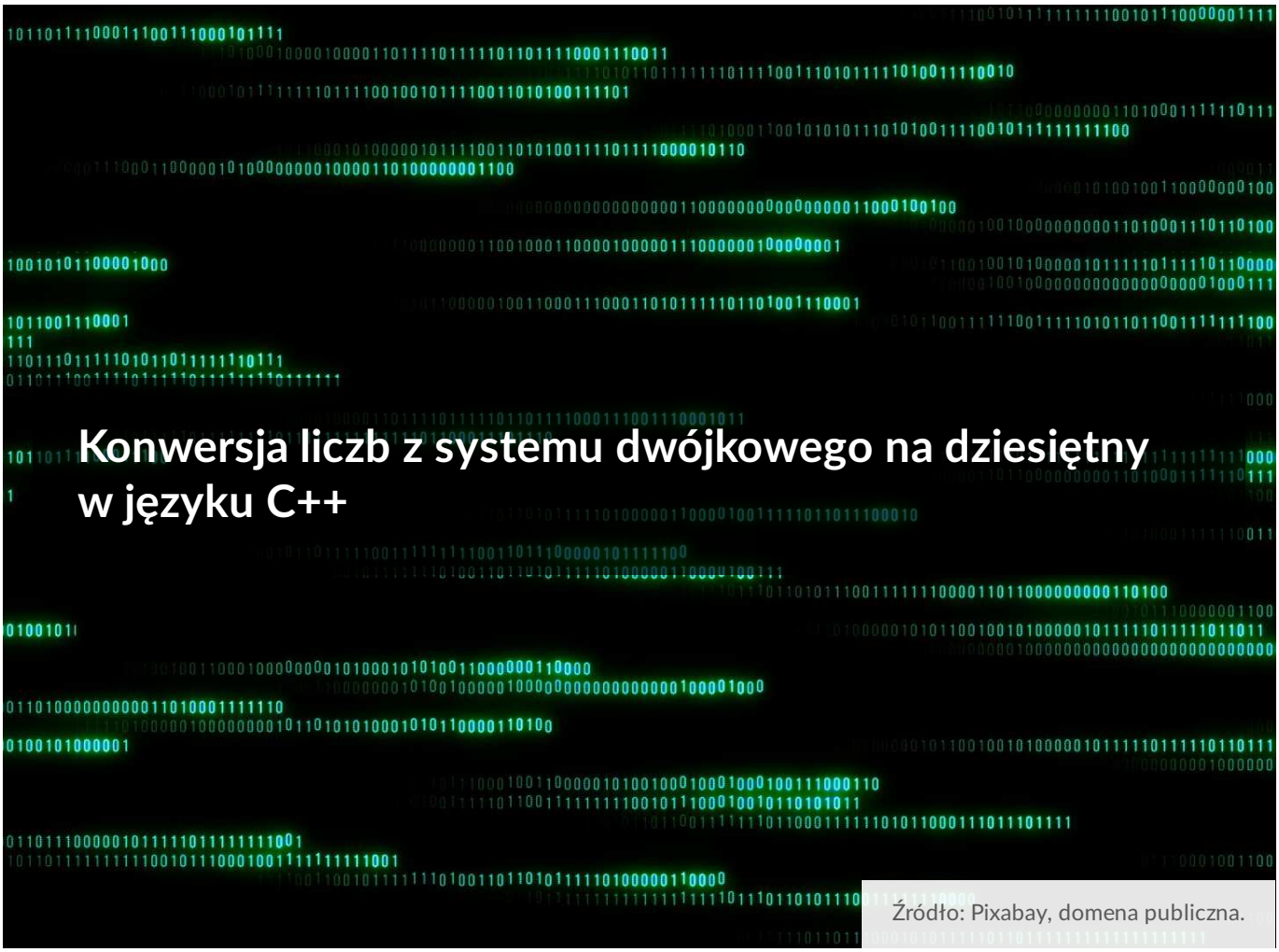


Konwersja liczb z systemu dwójkowego na dziesiętny w języku C++



Źródło: Pixabay, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Dziesiętny system liczbowy, którym posługujemy się na co dzień, jest wygodny w użyciu z punktu widzenia człowieka, a zarazem kłopotliwy do zastosowania we współczesnych układach cyfrowych, takich jak procesory lub pamięci. Działanie wielu urządzeń oparte jest na systemie binarnym (dwójkowym).

Poznaliśmy już algorytm konwersji liczb z systemu dwójkowego na dziesiętny. W tym e-materiale zaimplementujemy go 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:

- Konwersja liczb z systemu dwójkowego na dziesiętny w języku Java,
- Konwersja liczb z systemu dwójkowego na dziesiętny w języku Python.

Więcej zadań? Sięgnij do Konwersja liczb z systemu dwójkowego na dziesiętny – zadania maturalne.

> **Twoje cele**
>
> - Przeanalizujesz algorytm konwersji części ułamkowej liczby z systemu binarnego na dziesiętny.
> - Zaimplementujesz algorytm konwersji z systemu dwójkowego na dziesiętny w języku C++.
> - Scharakteryzujesz algorytm konwersji liczby zapisanej w systemie binarnym na liczbę w systemie dziesiętnym.

Przeczytaj

Zanim przystąpimy do implementacji algorytmu konwersji liczb z systemu dwójkowego na dziesiętny, przypomnijmy, w jaki sposób dokonywane jest takie przekształcenie. Przeanalizujmy przykład zamiany liczby zapisanej w systemie dwójkowym (binarnym) na system dziesiętny, czyli o podstawie 10. Ułatwi to późniejsze zaimplementowanie algorytmu w języku C++.

Dokonamy konwersji liczby 1011 z systemu dwójkowego na dziesiętny.

$$1011_{(2)} \rightarrow (\ )_{(10)}$$

Jak rozwiązalibyśmy to zadanie, gdybyśmy mieli do dyspozycji kartkę i długopis?

$$1011_{(2)} = (\ )_{(10)}$$
$$1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 =$$
$$1 + 2 + 0 + 8 = 11$$

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

Konwersja polega na mnożeniu kolejnych bitów (zaczynając od najmłodszego bitu i kończąc na najstarszym bicie) przez wagę pozycji, czyli kolejne potęgi liczby 2 (zaczynając od potęgi zerowej). Po zsumowaniu iloczynów częściowych otrzymamy wynik – liczbę zapisaną w systemie dziesiętnym.

Oto wynik końcowy wraz z odpowiednim zapisem:

$$1011_{(2)} \rightarrow (11)_{(10)}$$

Komputerowa realizacja w języku C++

Teraz zastanówmy się, jak zaimplementować omówiony algorytm przy użyciu języka programowania C++.

Algorytm zamiany liczby z systemu binarnego na system dziesiętny zapiszemy w ciele funkcji o nazwie `konwertujBinDec()`. Zwróci ona zmienną typu całkowitego (`int`) – będzie to liczba przekształcona do postaci dziesiętnej. Jako argument funkcja przyjmie liczbę zapisaną w systemie binarnym i przechowywaną w zmiennej typu `string liczbaBin`. Dlaczego użyjemy zmiennej typu `string`? W tym przypadku łatwiej będzie użyć właśnie tego typu zmiennej – podczas konwersji potrzebna jest znajomość wartości liczbowych poszczególnych cyfr, nie zaś całej liczby.

> **Ważne!**
>
> W przypadku operowania na zmiennych typu `string` pamiętajmy o użyciu odpowiedniej biblioteki:
>
> ```
> 1 #include <string>
> ```

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int konwertujBinDec(string liczbaBin) {
6
7 }
```

Następnie zadeklarujmy kilka zmiennych:

- `int wynik` – zmienna, w której będziemy sumować kolejne iloczyny częściowe powstałe wskutek mnożenia wag przez bity. Będzie to wartość zwracana przez funkcję `konwertujBinDec()`,
- `int waga` – będzie przyjmować kolejne wagi bitów, czyli kolejne potęgi liczby 2,
- `int bit` – do niej zapiszemy konkretny bit konwertowanej liczby.

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

int konwertujBinDec(string liczba_bin) {
    int wynik = 0;
    int waga = 1;
    int bit = 0;
}
```

Zmienna `waga` na początku przyjmuje wartość 1, ponieważ wagą najmłodszego bitu jest 2^0, czyli 1.

Napiszmy teraz pętlę typu `for`, w której kolejne bity, zaczynając od najmłodszego, będą mnożone przez odpowiadające im wagi.

```
int konwertujBinDec(string liczbaBin) {
    int wynik = 0;
    int waga = 1;
    int bit = 0;

    for (int i = liczbaBin.length() - 1; i >= 0; i--) {
        bit = liczbaBin[i] - '0';
        wynik = wynik + bit * waga;
        waga = waga * 2;
    }

    return wynik;
}
```

Omówmy teraz poszczególne fragmenty kodu.

- `int bit = liczbaBin[i] - '0'` → w zmiennej typu `int` zapisujemy zmienną typu `string` i odejmujemy od niej liczbę odpowiadającą w kodzie ASCII znakowi

zera. Dzięki temu w zmiennej `bit` zostaje zapisana liczba 0 lub 1. Zmienna `liczbaBin[i]` w kodzie ASCII to znak '0' lub '1' (ponieważ tylko takie liczby mogą pojawić się w systemie binarnym), a odpowiadające tym znakom liczby to, kolejno, 48 i 49. Oznacza to, że jeżeli `liczbaBin[i]` będzie znakiem '1', zostanie wykonane działanie $49 - 48$, czyli zmienna `bin` przyjmie wartość liczbową 1. Jeżeli natomiast `liczbaBin[i]` będzie znakiem '0', wykonane zostanie działanie $48 - 48$, a zmienna `bin` przyjmie wartość liczbową 0,

- `wynik = wynik + bit * waga` → dodajemy kolejne iloczyny częściowe,
- `waga = waga * 2` → aby uzyskać wagę kolejnej pozycji, mnożymy aktualną wagę razy podstawę systemu, czyli 2.

Napiszmy kod, który umożliwi użytkownikowi wpisanie własnej liczby w celu wykonania konwersji do systemu dziesiętnego. Skorzystamy ze strumienia `cin`. Przeanalizujmy poniższy fragment kodu:

```
string liczbaTest;

cin >> liczbaTest;
```

Oczywiście musimy zadeklarować zmienną `liczbaTest`, do której zapisana zostanie informacja podana przez użytkownika.

Kolejnym krokiem będzie wywołanie funkcji `konwertujBinDec()` dla argumentu `liczbaTest` i wypisanie wyniku (czyli liczby przekształconej do postaci dziesiętnej). W tym przypadku posłużymy się strumieniem `cout`.

Oto kod całego programu:

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

int konwertujBinDec(string liczbaBin) {
```

```
    int wynik = 0;
    int waga = 1;
    int bit = 0;

    for (int i = liczbaBin.length() - 1; i >= 0; i--) {
        bit = liczbaBin[i] - '0';
        wynik = wynik + bit * waga;
        waga = waga * 2;
    }

    return wynik;
}

int main() {
    string liczbaTest;

    cin >> liczbaTest;
    cout << konwertujBinDec(liczbaTest);

    return 0;
}
```

Alternatywnym rozwiązaniem powyższego problemu jest zastosowanie schematu Hornera przy konwersji. Pozwoli to zmniejszyć złożoność obliczeniową algorytmu do złożoności liniowej.

Poniższej przedstawiona została implementacja rozwiązania alternatywnego:

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

int konwertujBinDec(string liczbaBin) {
    int wynik = 0;
```

```
    int bit = 0;

    for (int i = 0; i < liczbaBin.size(); i++) {
        bit = liczbaBin[i] - '0';
        wynik = wynik * 2 + bit;
    }

    return wynik;
}

int main() {
    string liczbaTest;

    cin >> liczbaTest;
    cout << konwertujBinDec(liczbaTest);

    return 0;
}
```

Konwersja części ułamkowej

Konwersja części ułamkowej z systemu binarnego na dziesiętny przebiega bardzo podobnie do przekształcania części całkowitej. Mnożymy kolejne bity przez wagi pozycji, jednak tym razem są one kolejnymi potęgami odwrotności podstawy systemu, czyli liczby 2.

Posłużmy się przykładem:

$$0{,}1101_{(2)} = (\quad)_{(10)}$$

$$1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$

$$0{,}5 \quad + 0{,}25 \quad + \quad 0 \quad + 0{,}0625 = 0{,}8125$$

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

Mnożymy kolejne bity (zaczynając od bitu umieszczonego skrajnie po lewej stronie) przez wagę pozycji, czyli przez kolejne odwrotności potęgi liczby 2 (zaczynając od 2^{-1}). Następnie sumujemy iloczyny częściowe. Tak otrzymany wynik jest ułamkiem zapisanym w systemie dziesiętnym.

Słownik

> **najmłodszy bit**
>
> bit o najmniejszej wadze, najbardziej wysunięty na prawo

> **najstarszy bit**
>
> bit o największej wadze, najbardziej wysunięty na lewo

Symulacja interaktywna

Polecenie 1

Przeanalizuj prezentację implementacji algorytmu konwersji części ułamkowej liczby z systemu binarnego na dziesiętny.

1

Napiszmy program konwertujący część ułamkową liczby w systemie dwójkowym na system dziesiętny. Zakładamy, że liczba poddawana konwersji będzie mniejsza od 1.

2

Zacznijmy od zbudowania szkieletu programu, czyli zadeklarowania głównej funkcji - `main()`. W jej ciele umieścimy niezbędne instrukcje.

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

int main() {
    return 0;
}
```

3

Stwórzmy szkielet funkcji `konwertujUlamek()`. Zwróci ona zmienną typu `double`. Będzie w niej zapisany wynik konwersji, czyli liczba ułamkowa w postaci dziesiętnej. Pamiętajmy o bibliotece `iostream`.

```
#include <iostream>
```

```
using namespace std;

double konwertujUlamek() {

}

int main() {
    return 0;
}
```

4

Argumentem, który funkcja będzie przyjmowała na wejście, jest zmienna `ulamekBin`, czyli liczba przeznaczona do przekonwertowania. Jest to argument typu `string`.

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

double konwertujUlamek(string ulamekBin) {

}

int main() {
    return 0;
}
```

5

Zadeklarujmy dwie zmienne. Pierwszą z nich jest `int bit`. Ustalimy jej wartość początkową na 0. Później zmienna ta będzie przyjmować wartość 0 lub 1, w zależności od odczytanego znaku.

Drugą zadeklarowaną zmienną jest `double waga`. W kolejnych iteracjach przyjmie ona wartości odpowiadające potęgom odwrotności liczby 2. Wartością początkową jest 0,5, czyli 2^{-1}.

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

double konwertujUlamek(string ulamekBin) {
    int bit = 0;
    double waga = 0.5;
}

int main() {
    return 0;
}
```

6

Nie zapominajmy o zadeklarowaniu zmiennej odpowiedzialnej za zapisywanie wyniku. Musi być ona typu `double`.

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

double konwertujUlamek(string ulamekBin) {
    int bit = 0;
    double waga = 0.5;
    double wynik = 0;
}

int main() {
    return 0;
}
```

7

Utwórzmy teraz pętlę `for`. Będziemy dzięki niej odczytywać kolejne elementy składające się na część ułamkową liczby binarnej.

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

double konwertujUlamek(string ulamekBin) {
    int bit = 0;
    double waga = 0.5;

    for (int i = 2; i < ulamekBin.length(); i++) {

    }
}

int main() {
    return 0;
}
```

Zwróćmy uwagę na to, że w pętli `for` odczytywanie ułamka zaczyna się od elementu `ulamekBin[2]`. Dzieje się tak, ponieważ pierwsze dwa znaki zmiennej to zero i przecinek, czyli `ulamekBin[0]` i `ulamekBin[1]` (odpowiednio, '0' oraz ',').

8

Przejdźmy do implementacji algorytmu konwersji.

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

```
4
5  double
   konwertujUlamek(string
   ulamekBin) {
6      int bit = 0;
7      double waga = 0.5;
8
9      for (int i = 2; i <
   ulamekBin.length(); i++) {
10         bit = ulamekBin[i]
   - '0';
11         wynik = wynik +
   bit * waga;
12         waga = waga * 0.5;
13     }
14
15     return wynik;
16 }
17
18 int main() {
19     return 0;
20 }
```

Funkcja zwraca zmienną `wynik`, czyli ułamek przekształcony z systemu binarnego do postaci dziesiętnej.

Program jest już prawie gotowy. Dodajmy polecenia odpowiadające za interakcję z użytkownikiem, czyli umożliwiające wpisanie liczby do przekonwertowania.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  double
   konwertujUlamek(string
   ulamekBin) {
6      int bit = 0;
7      double waga = 0.5;
```

```
    double wynik = 0;

    for (int i = 2; i < ulamekBin.length(); i++) {
        bit = ulamekBin[i] - '0';
        wynik = wynik + bit * waga;
        waga = waga * 0.5;
    }

    return wynik;
}

int main() {
    string ulamekTest;
    cin >> ulamekTest;

    return 0;
}
```

10

Na koniec musimy dodać wywołanie funkcji `konwertujUlamek()` dla parametru `ulamekTest` i wypisać wynik.

Oto gotowy program:

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

double konwertujUlamek(string ulamekBin) {
    int bit = 0;
    double waga = 0.5;
    double wynik = 0;

    for (int i = 2; i < ulamekBin.length(); i++) {
```

```
        bit = ulamekBin[i] - '0';
        wynik = wynik + bit * waga;
        waga = waga * 0.5;
    }

    return wynik;
}

int main() {
    string ulamekTest;

    cin >> ulamekTest;
    cout <<"wynik: "<< konwertujUlamek(ulamekTest);

    return 0;
}
```

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

Polecenie 2

Zapoznaj się z symulacją interaktywną prezentującą konwersję liczb z systemu binarnego na system dziesiętny.

Zasób interaktywny dostępny pod adresem https://zpe.gov.pl/a/D12gitIoQ

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

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Poniższy kod powinien być implementacją algorytmu konwersji części całkowitej liczby z sytemu binarnego na dziesiętny. Kod jest niepełny. Uzupełnij blok pętli `for` i uruchom program dla wprowadzonych danych.

Specyfikacja problemu:

Dane:

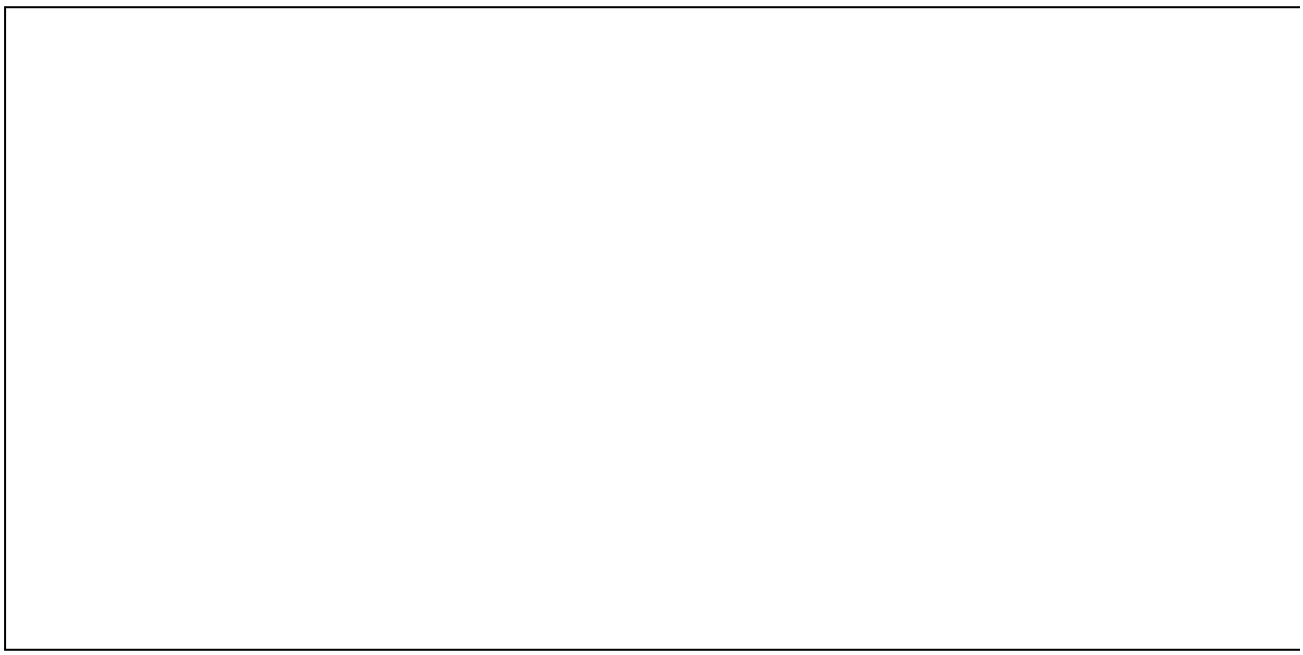
- `liczba` – liczba zapisana w systemie binarnym; zmienna typu *string*

Wynik:

Program wyświetla liczbę po konwersji z systemu binarnego na system dziesiętny.

Twoje zadania

1. Program ma konwertować liczbę daną liczbę zapisaną w systemie binarnym do postaci dziesiętnej.



Ćwiczenie 2

Napisz program, który dokona konwersji ułamka zapisanego w systemie binarnym na system dziesiętny. Przetestuj działanie programu dla ułamka 0,11101.

Specyfikacja problemu:

Dane:

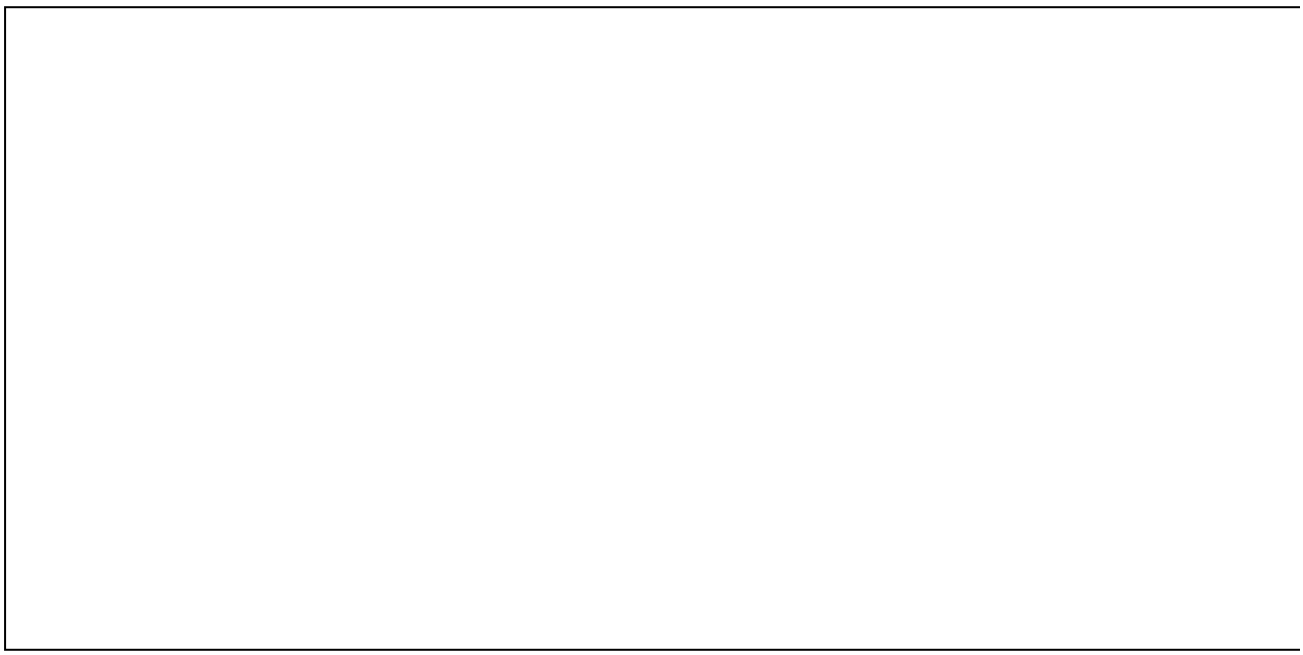
- `ulamek` – ułamek zapisany w systemie binarnym; zmienna typu *string*

Wynik:

Program wypisuje wynik konwersji ułamka zapisanego w systemie binarnym na system dziesiętny.

Twoje zadania

1. Program ma konwertować ułamek zapisany za pomocą systemu binarnego do postaci dziesiętnej.



Ćwiczenie 3

Napisz program, który przekształci liczbę dwójkową do postaci dziesiętnej. Przetestuj działanie programu dla liczby 1001.0001.

Specyfikacja problemu:

Dane:

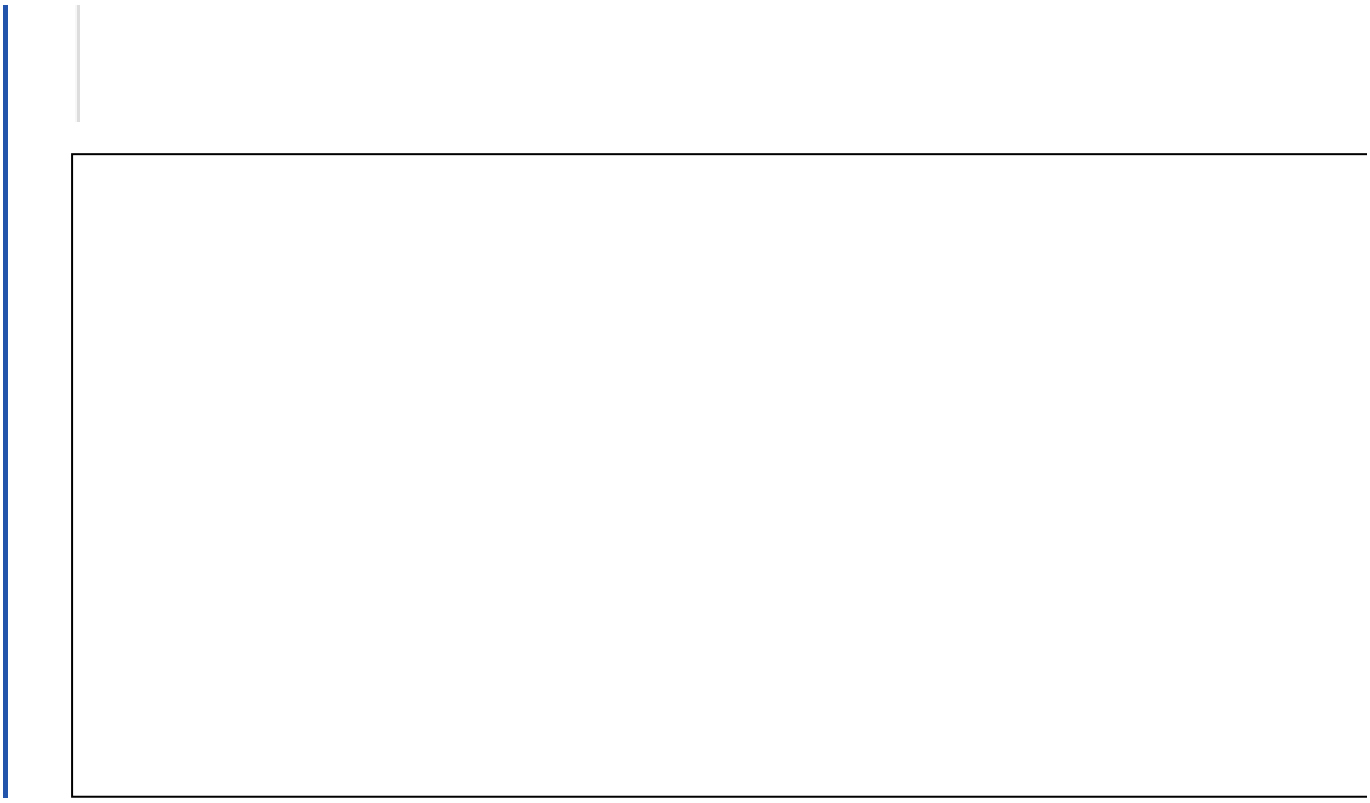
- `liczbaBin` – liczba zapisana w systemie dwójkowym; zmienna typu *string*

Wynik:

Program wyświetla wynik konwersji liczby zapisanej w systemie binarnym na liczbę zapisaną w systemie dziesiętnym.

Twoje zadania

1. Program ma konwertować liczbę dwójkową do postaci dziesiętnej.



Dla nauczyciela

Autor: Maurycy Gast

Przedmiot: Informatyka

Temat: Konwersja liczb z systemu dwójkowego na dziesiętny w języku C++

Grupa docelowa:

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

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:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

a) na liczbach: badania pierwszości liczby, zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, działań na ułamkach z wykorzystaniem NWD i NWW,

b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara i przestawieniową,

c) porządkowania ciągu liczb: przez wstawianie i metodą bąbelkową,

d) wydawania reszty najmniejszą liczbą nominałów,

e) obliczania wartości elementów ciągu metodą iteracyjną i rekurencyjną, w tym wartości elementów ciągu Fibonacciego.

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 konwersji części ułamkowej liczby z systemu binarnego na dziesiętny.
- Zaimplementujesz algorytm konwersji z systemu dwójkowego na dziesiętny w języku C++.
- Scharakteryzujesz algorytm konwersji liczby zapisanej w systemie binarnym na liczbę w systemie dziesiętnym.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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: „Konwersja liczb z systemu dwójkowego na dziesiętny w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel prosi wybraną osobę o odczytanie tematu lekcji, a następnie określa cele i kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
   – co to jest najmłodszy bit?
   – co to jest najstarszy bit?

– w jaki sposób dokonywana jest konwersja liczb z systemu dwójkowego na dziesiętny?
– jak przebiega konwersja części ułamkowej z systemu binarnego na dziesiętny?
Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. Uczniowie analizują przykład z sekcji „Przeczytaj" i powtarzają zaprezentowane rozwiązanie na swoim komputerze.
2. Uczniowie w parach wykonują ćwiczenia nr 1–4. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku C++.

Praca domowa:

1. Uczniowie opracowują FAQ (minimum 3 pytania i odpowiedzi) do tematu lekcji („Konwersja liczb z systemu dwójkowego na dziesiętny w języku C++").
2. Uczniowie wykonują polecenia 1 i 2 z sekcji „Symulacja interaktywna".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- 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:

- Uczniowie mogą wykorzystać multimedium w sekcji „Symulacja interaktywna” do przygotowania się do lekcji powtórkowej.