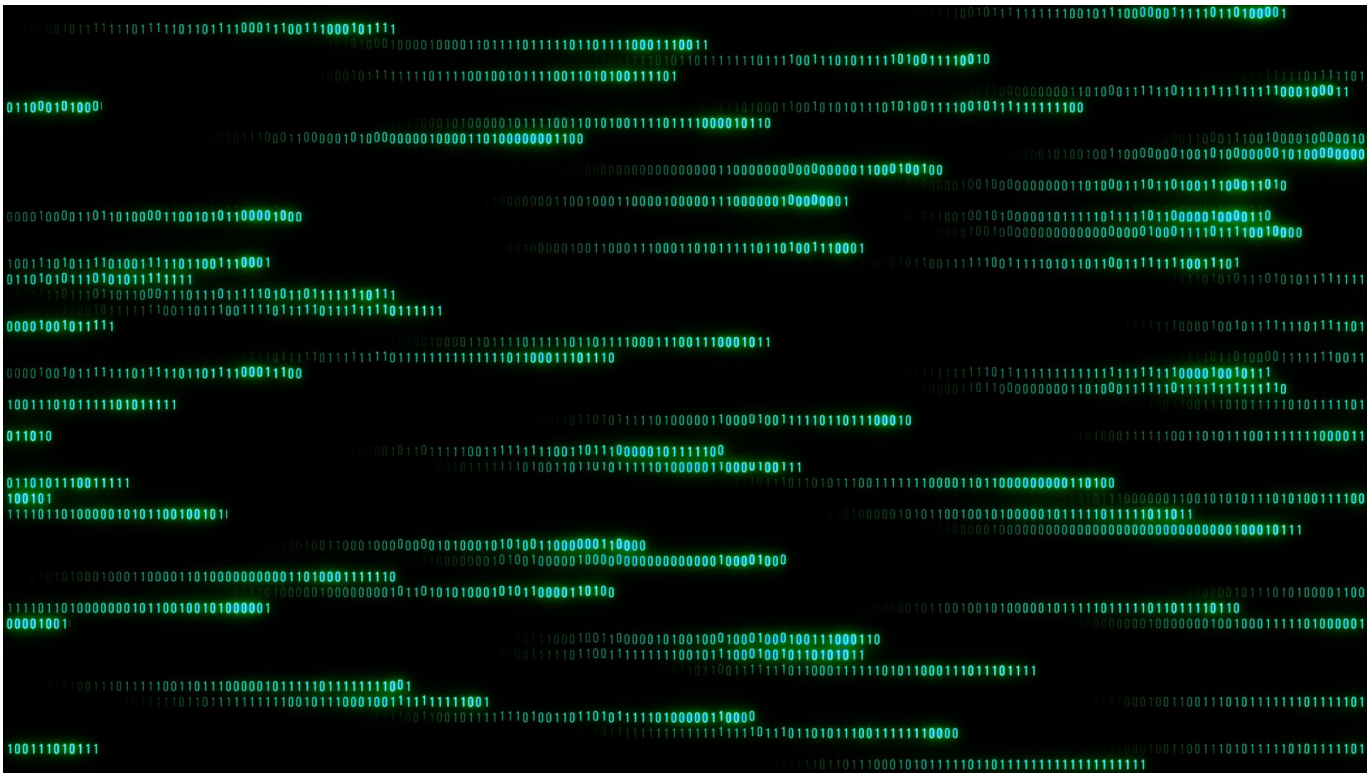
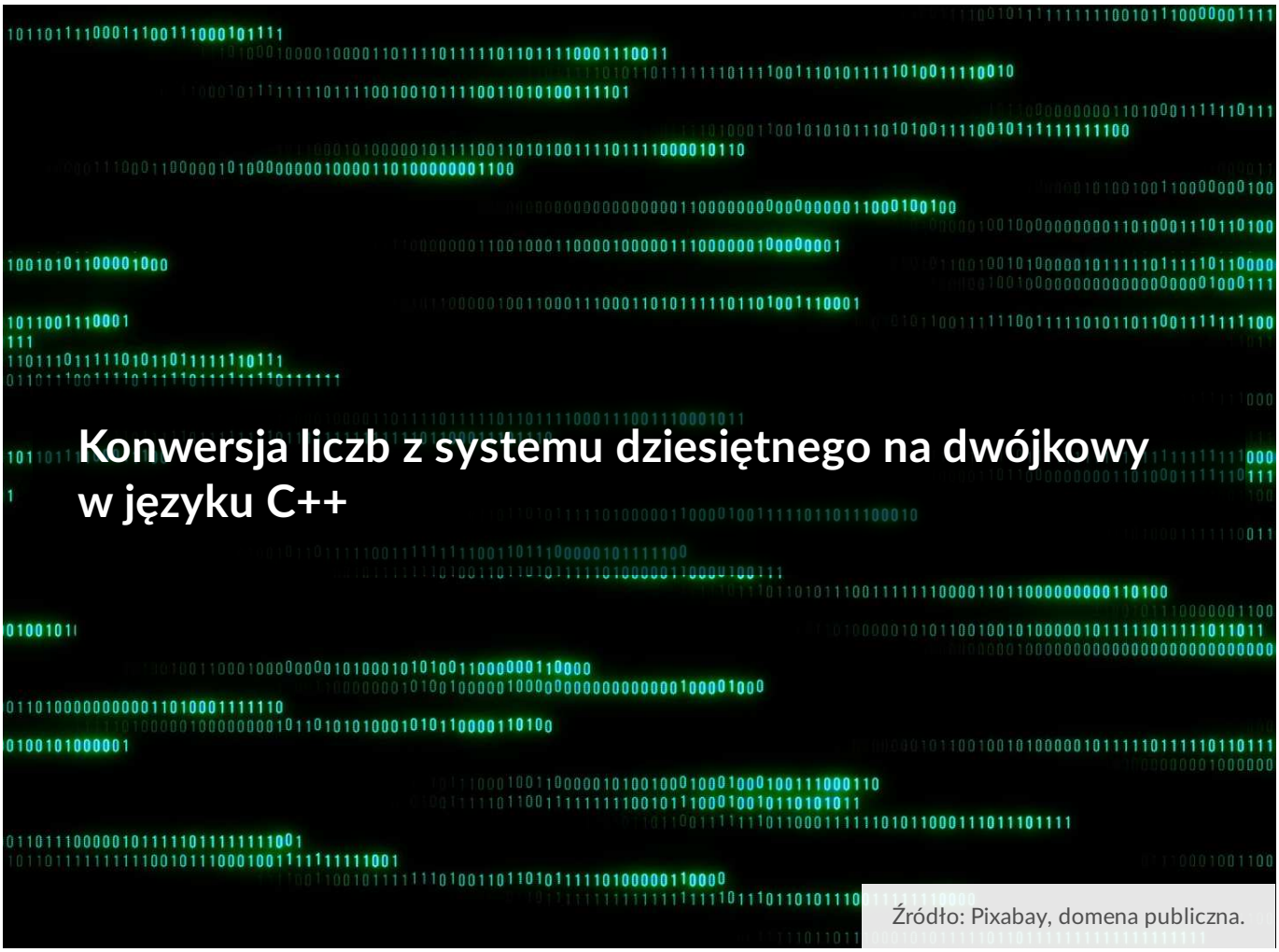


Konwersja liczb z systemu dziesiętnego na dwójkowy w języku C++



Źródło: Pixabay, domena publiczna.

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

Poznaliśmy już algorytm konwersji liczb z systemu dziesiętnego na dwójkowy (binarny). Przedstawiliśmy ręczną metodę przeprowadzania takiej zamiany oraz pseudokod opisujący niezbędne czynności.

Ten e-materiał poświęcimy zaimplementowaniu poznanego algorytmu 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 dziesiętnego na dwójkowy w języku Java,
- Konwersja liczb z systemu dziesiętnego na dwójkowy w języku Python.

Więcej zadań? Sięgnij do Konwersja liczb z systemu dziesiętnego na dwójkowy – zadania maturalne.

> **Twoje cele**
>
> - Zaimplementujesz w języku C++ algorytm konwersji części całkowitej liczby dziesiętnej do postaci binarnej.

- Poznasz sposób zamiany części ułamkowej liczby dziesiętnej do postaci dwójkowej i zastosujesz go w samodzielnie napisanym programie.
- Wykonasz kilka zadań związanych z konwersją liczb między systemami o różnych podstawach.

Film samouczek

Problem 1

Specyfikacja problemu:

Napisz konwerter liczb z systemu dziesiętnego na dwójkowy (binarny).

Polecenie 1

Porównaj swoje rozwiązanie z przedstawionym w filmie.

Film dostępny pod adresem /preview/resource/R1dYCakE8ZZyo

Film nawiązujący do treści lekcji dotyczący konwersji liczby dziesiętnej na dwójkową w języku C++.

Przeczytaj

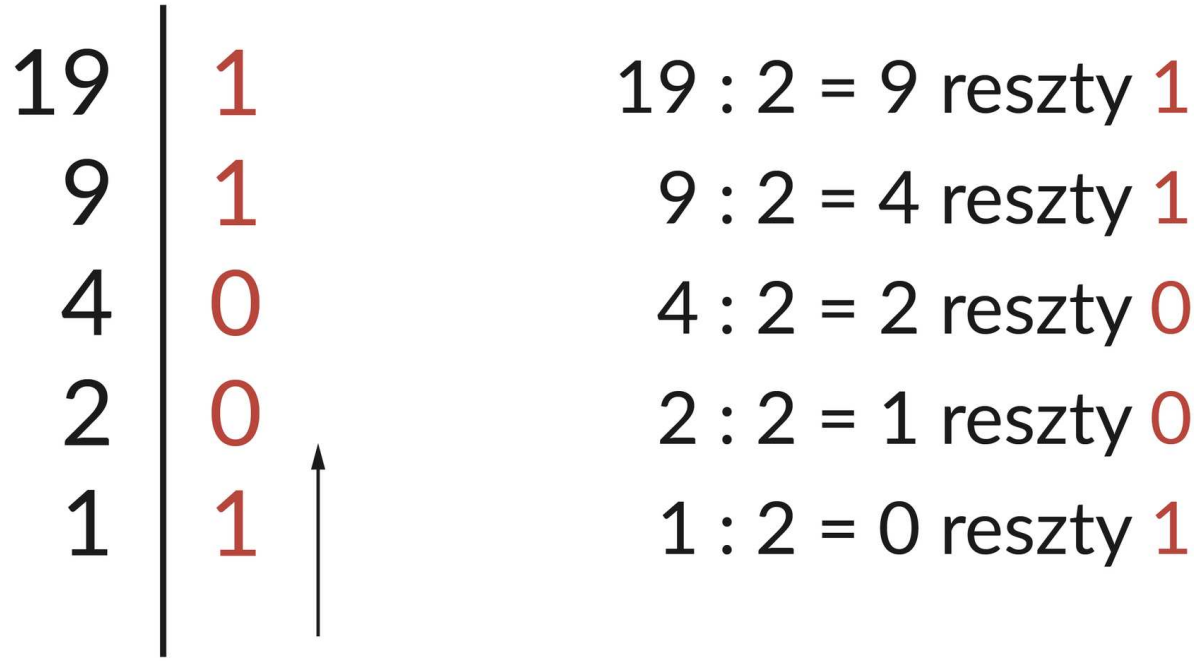
Konwersja części całkowitej liczby

Przypomnijmy sobie algorytm konwersji liczb zapisanych w postaci dziesiętnej do ich dwójkowych odpowiedników.

Zamienimy liczbę 19 na postać dwójkową:

$$19_{(10)} \quad \rightarrow \quad (\ \ )_{(2)}$$

Zadanie takie wykonamy, dzieląc wielokrotnie liczbę 19 przez 2 i zapisując resztę pozostałą po operacji.



Kolejne etapy przekształcania liczby zapisanej w systemie dziesiętnym do postaci dwójkowej
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Dzielnik ma wartość 2, ponieważ taka jest baza (podstawa) docelowego systemu liczbowego. Prześledźmy wykonywane czynności.

1. Liczbę 19 dzielimy przez 2; zapisujemy całkowitą część wyniku (czyli 9) i resztę z dzielenia (wynosi ona 1).
2. Dzielimy przez 2 część całkowitą z poprzedniej operacji (9). Po raz kolejny zapamiętujemy iloraz (4) i resztę (ponownie wynosi ona 1).
3. Znowu dzielimy ostatni otrzymany iloraz przez 2, zapamiętując część całkowitą wyniku i resztę.
4. Powtarzamy przedstawione operacje do momentu, w którym iloraz wyniesie zero.

5. Otrzymane w kolejnych etapach reszty z dzielenia, ustawione w kolejności od ostatniej do pierwszej, składają się na dwójkową postać przekształcanej liczby. Na rysunku zaznaczyliśmy reszty kolorem czerwonym. Cyfry składające się na wynik odczytujemy **od dołu do góry**, tak jak pokazuje strzałka.
6. Otrzymujemy wynik:

$$19_{(10)} = 10011_{(2)}$$

Zajmijmy się teraz zapisaniem algorytmu w języku C++.

Algorytm konwersji w języku C++

Pisząc w języku C++ program dokonujący konwersji liczby dziesiętnej do postaci binarnej, wykorzystamy:

- **samodzielnie zdefiniowaną funkcję** – będzie ona odpowiadać za cały proces zmiany postaci liczby;
- **strumienie wejścia/wyjścia** – użytkownik poda dzięki nim liczbę przeznaczoną do zapisania w postaci dwójkowej, a później zobaczy wynik działania programu;
- **funkcje wbudowane języka C++** – posłużymy się nimi podczas przekształcania liczby i prezentacji rezultatów.

Najpierw pozwolimy użytkownikowi podać liczbę przeznaczoną do zapisania w systemie binarnym. Wykorzystamy strumień wejścia `cin`:

```
int liczba_dec;

std::cin >> liczba_dec;
```

W tej części kodu definiujemy zmienną całkowitą `liczba_dec`. Przyjmie ona wartość liczby, którą wpisał użytkownik.

Następnie deklarujemy funkcję o nazwie `konwertuj()`. Właśnie ona zajmie się zamianą postaci liczby:

```
#include <string>

std::string konwertuj(int liczba) {

}
```

Jako argument funkcja przyjmie liczbę całkowitą, natomiast zwróci ciąg znaków (obiekt typu `string`). Decydujemy się na taki właśnie typ danych ze względu na wygodę. Łańcucha znaków będzie nam po prostu łatwo używać.

Pamiętajmy, że proces konwersji polega na „sklejaniu" w całość kolejnych liczb będących resztami z dzielenia przekształcanej liczby przez 2. Język C++ pozwala wykonywać takie zadania bardzo łatwo, kiedy operuje się na łańcuchach znaków.

Liczbę w postaci dwójkowej przedstawimy jako ciąg znaków o nazwie `bin`. Początkowo będzie on pusty, ale wraz z każdym cyklem dzielenia dopiszemy do niego wartość 0 lub 1 (czyli kolejne reszty). W przypadku obiektów typu `string` podczas „doklejania" elementów do końca ciągu używa się zwykłego operatora dodawania.

W końcowej fazie działania programu wyświetlimy zmienną `bin` wspak (podobnie jak w zapisie ręcznym podawaliśmy wynik od dołu do góry). Również taką operację przeprowadza się na ciągach znaków bardzo łatwo.

Dopiszmy zatem niezbędne instrukcje w ciele funkcji `konwertuj()`:

```
std::string konwertuj(int liczba) {
    std::string bin = "";
    int reszta = 0;

    while (liczba > 0) {
        reszta = liczba % 2;
        liczba = liczba / 2;
        bin = bin + std::to_string(reszta);
    }

    std::string bin_result = "";

    for (int i = bin.length() - 1; i >= 0; i--) {
        bin_result += bin[i];
    }

    return bin_result;
}
```

Zauważ, że w programie użyliśmy funkcji wbudowanej `to_string()`. Musimy się nią posłużyć, aby uniknąć błędów podczas kompilacji.

Zmienna `bin` jest typu `string`, dopisujemy do niej („doklejamy") zmienną typu całkowitego `reszta`. Aby je połączyć, nie wolno tak po prostu użyć znaku dodawania. Oznaczałoby to próbę arytmetycznego dodania liczby 0 lub 1 do ciągu znaków. Kompilator nie pozwoli tego zrobić ze względu na niezgodność typów danych.

Właśnie dlatego sięgnęliśmy po funkcję `to_string()`. Przekształca ona typ zmiennej `reszta` w ciąg znaków (`string`). W efekcie możemy skorzystać z operatora `+` aby dopisać 0 lub 1 na końcu ciągu `bin`.

W ostatniej pętli zapisujemy w zmiennej `bin_result` ciąg znaków `bin` w kolejności od ostatniego do pierwszego, a rezultat wyświetlamy na ekranie.

Oto cały program dokonujący konwersji liczby zapisanej w systemie dziesiętnym do postaci binarnej:

```
#include <iostream>
#include <string>

std::string konwertuj(int liczba) {
    std::string bin = "";
    int reszta = 0;

    while (liczba > 0) {
        reszta = liczba % 2;
        liczba = liczba / 2;
        bin = bin + std::to_string(reszta);
    }

    std::string bin_result = "";

    for (int i = bin.length() - 1; i >= 0; i--) {
        bin_result += bin[i];
    }

    return bin_result;
}


int main() {
    int liczba_dec;

    std::cin >> liczba_dec;
```

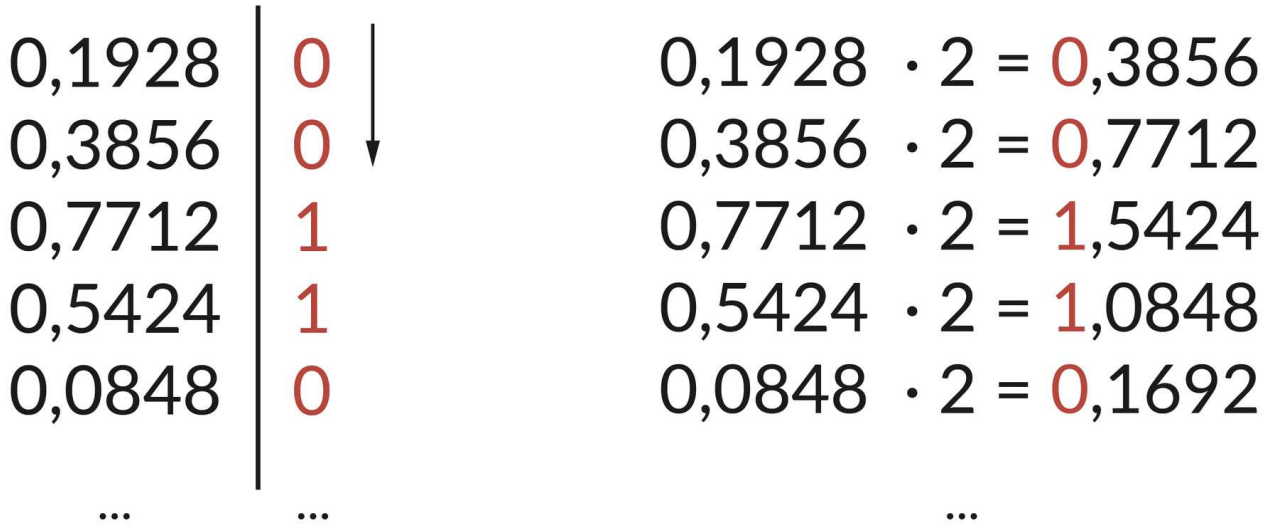
```
28      std::cout << konwertuj(liczba_dec);
29
30      return 0;
31  }
```

Konwersja części ułamkowej

W przypadku konwersji części ułamkowej z systemu dziesiętnego do postaci binarnej będziemy postępować inaczej niż podczas przekształcania części całkowitej.

Jeszcze raz posłużymy się przykładem. Zapiszemy liczbę 0,1928 w postaci dwójkowej.

$$0,1928_{(10)} \quad \rightarrow \quad (\quad)_{(2)}$$



Przekształcanie części ułamkowej liczby do postaci binarnej
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W tym przypadku wielokrotnie **mnożymy** część ułamkową przez liczbę 2 (czyli podstawę systemu dwójkowego) i sprawdzamy, czy iloczyn jest większy niż 1.

Wróćmy do przykładu:

1. Mnożymy liczbę 0,1928 przez 2.
2. Sprawdzamy wartość iloczynu:
    - jeżeli jest ona większa niż 1, przypisujemy cyfrze części ułamkowej wartość 1;
    - jeżeli jest ona mniejsza niż 1, przypisujemy cyfrze części ułamkowej wartość 0.
3. Jeżeli iloczyn był większy od jedności, odejmujemy od niego 1 (tak, aby została tylko część ułamkowa); w sytuacji, gdy iloczyn był mniejszy niż 1, pozostawiamy go w postaci niezmienionej.
4. Wracamy do punktu 1.; tym razem jako część ułamkową podstawiamy wartość otrzymaną w punkcie 3.

Kolejne cyfry części ułamkowej poznajemy w punkcie 2. przedstawionego algorytmu. Tym razem jednak odczytujemy je nie wspak (jak w przypadku konwersji części całkowitej), lecz wprost – **od góry do dołu**. A zatem:

$$0,1928_{(10)} \quad \rightarrow \quad (0,00110)_{(2)}$$

Słownik

system binarny

inna nazwa systemu dwójkowego (czyli systemu liczbowego o podstawie 2), w którym wykorzystywane są tylko cyfry 1 i 0

baza

synonim podstawy pozycyjnego systemu liczbowego; przykładowo, bazą systemu ósemkowego jest liczba 8

Prezentacja multimedialna

Polecenie 1

Przeanalizuj implementację algorytmu konwersji części ułamkowej liczby dziesiętnej do postaci binarnej.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

W poprzedniej sekcji został omówiony sposób konwertowania części ułamkowej liczby. Pisząc program, skorzystamy z tej wiedzy.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Zaczniemy od zadeklarowania głównej funkcji programu - `main`. Dołączymy też bibliotekę `<iostream>`, dzięki czemu będziemy mogli komunikować się z użytkownikiem:

```
#include <iostream>

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

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Następnie zadeklarujemy funkcję `konwertuj_ulamek()`. Zwróci ona zmienną typu `string`. Będzie to wynik konwersji, czyli liczba ułamkowa zapisana w systemie dwójkowym. Pamiętajmy o dołączeniu biblioteki `<string>`:

```
#include <iostream>
#include <string>

std::string
konwertuj_ulamek() {

}

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

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Jednym z argumentów funkcji będzie zmienna `liczba`. Właśnie ona zostanie zapisana w postaci binarnej. Ten argument jest typu `double`:

```
#include <iostream>
#include <string>

std::string
konwertuj_ulamek(double
liczba) {

}

int main() {
    return 0;
```

```
10 }
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Drugi argument to zmienna typu całkowitego `precyzja`. Użytkownik poinformuje dzięki niej program, do ilu miejsc po przecinku chce zapisać liczbę dwójkowej:

```
1  #include <iostream>
2  #include <string>
3
4  std::string
   konwertuj_ulamek(double
   liczba, int precyzja) {
5
6  }
7
8  int main() {
9      return 0;
10 }
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

W ciele funkcji zdefiniujemy zmienną `precyzja_start` i przypiszemy jej wartość 0. Będzie się ona zwiększać aż do momentu osiągnięcia żądanej precyzji (czyli podanej przez użytkownika liczby miejsc po przecinku). Zadeklarujemy też zmienną `std::string wynik`. Właśnie ona ma przechować rezultat konwersji, czyli część ułamkową w postaci dwójkowej:

```
#include <iostream>
#include <string>

std::string konwertuj_ulamek(double liczba, int precyzja) {
    int precyzja_start = 0;
    std::string wynik = "0,";
}

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

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Kolejną czynnością jest utworzenie pętli `while`. Będzie ona wykonywana do momentu, w którym osiągniemy wskazaną przez użytkownika precyzję:

```
#include <iostream>
#include <string>

std::string konwertuj_ulamek(double liczba, int precyzja) {
    int precyzja_start = 0;
    std::string wynik = "0,";

    <span lang='en'>while</span> (precyzja_start < precyzja) {

```

```
    }
}

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

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Pora wykorzystać algorytm konwersji omówiony w poprzedniej sekcji:

```
#include <iostream>
#include <string>

std::string konwertuj_ulamek(double liczba, int precyzja) {

    int precyzja_start = 0;
    std::string wynik = "0,";

    while (precyzja_start < precyzja) {
        liczba = liczba * 2;

        if (liczba > 1) {
            liczba = liczba - 1;
            wynik = wynik + "1";
        } else {
            wynik = wynik + "0";
        }
    }
```

```
}

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

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

W czasie działania programu inkrementujemy zmienną `precyzja_start`, aby pętla nie była wykonywana w nieskończoność:

```
#include <iostream>
#include <string>

std::string konwertuj_ulamek(double liczba, int precyzja) {

    int precyzja_start = 0;
    std::string wynik = "0,";

    while (precyzja_start < precyzja) {
        liczba = liczba * 2;

        if (liczba > 1) {
            liczba = liczba - 1;
            wynik = wynik + "1";
        } else {
            wynik = wynik + "0";
        }

```

```
        precyzja_start++;
    }
}

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

Instrukcja `return wynik` jest poleceniem zwrócenia rezultatu działania funkcji, czyli binarnej postaci części ułamkowej liczby.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1C8M5yJh

Pozostaje już tylko dodać kod pozwalający aplikacji komunikować się z użytkownikiem. Program poprosi o podanie liczby do przekształcenia do postaci dwójkowej oraz o ustalenie precyzji, a na koniec wyświetli wynik konwersji.

Oto wersja końcowa programu:

```
#include <iostream>
#include <string>

std::string konwertujUlamek(double liczba, int precyzja) {

    int precyzjaStart = 0;
    std::string wynik = "0,";

    while (precyzjaStart < precyzja) {
        liczba = liczba * 2;

        if (liczba >= 1) {
```

```
            liczba = liczba - 1;
            wynik = wynik + "1";
        } else {
            wynik = wynik + "0";
        }

        precyzjaStart++;
    }

    return wynik;
}

int main(int argc, const char * argv[]) {
    double liczbaTest;
    int precyzja;

    std::cout << "Podaj liczbę do przekonwertowania na system dwójkowy" <<std::endl;
    std::cin >> liczbaTest;
    std::cout << "Podaj do ilu miejsc po przecinku chcesz aby program przekonwertował liczbę" << std::endl;
    std::cin >> precyzja;
    std::cout << "Wynik: " << konwertujUlamek(liczbaTest, precyzja) << std::endl;

    return 0;
}
```

Polecenie 2

Specyfikacja problemu:

Napisz program konwertujący część ułamkową liczby dziesiętnej do postaci binarnej.

```
#include <iostream>
#include <string>

std::string konwertujUlamek(double liczba, int precyzja) {
    // tutaj dopisz kod
}

int main(int argc, const char * argv[]) {
    double liczbaTest = 20;
    int precyzja = 2; // dla testu zmień liczbę i precyzję

    std::cout << "Wynik: " << konwertujUlamek(liczbaTest, precyzja) << std::endl;

```

```
```

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Przedstawiony program powinien konwertować część całkowitą liczby z systemu dziesiętnego do dwójkowego. Znalazł się w nim jednak błąd i program podaje złe wyniki. Popraw odpowiednie instrukcje.

Twoje zadania

1. Usuń błąd z programu konwertującego liczbę 19 do postaci dwójkowej. Wskazówka: błąd znalazł się w deklaracji pętli `for`.

Ćwiczenie 2

Przedstawiony program powinien konwertować część ułamkową liczby z systemu dziesiętnego do dwójkowego. Znalazł się w nim jednak błąd i program podaje złe wyniki. Popraw wadliwe instrukcje.

Twoje zadania

1. Program powinien konwertować część ułamkową liczby zapisanej w systemie dziesiętnym do postaci dwójkowej. W kodzie znalazł się jednak błąd, który trzeba usunąć. Wskazówka: błąd pojawił się wewnątrz pętli `while`.

Ćwiczenie 3

Napisz program, który przekonwertuje podaną liczbę większą od jeden z systemu dziesiętnego na binarny. Użyj w programie obu poznanych funkcji: do konwersji liczby całkowitej i do konwersji części ułamkowej. Opracuj sposób rozbicia liczby na część całkowitą i część ułamkową; pomocna może być funkcja podłogi `floor`.

Twoje zadania

1. Program ma dokonać konwersji liczby 21.6875 do postaci binarnej o określonej precyzji. Wynik powinien zostać wydrukowany na standardowe wyjście.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Konwersja liczb z systemu dziesiętnego na dwójkowy w języku C++

Grupa docelowa:

Szkoła ponadpodstawowa, 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.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

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,

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje

poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

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):

- Zaimplementujesz w języku C++ algorytm konwersji części całkowitej liczby dziesiętnej do postaci binarnej.
- Poznasz sposób zamiany części ułamkowej liczby dziesiętnej do postaci dwójkowej i zastosujesz go w samodzielnie napisanym programie.
- Wykonasz kilka zadań związanych z konwersją liczb między systemami o różnych podstawach.

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: „Konwersja liczb z systemu dziesiętnego na dwójkowy w języku C++". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Film samouczek".

Faza wstępna:

1. Prowadzący wyświetla na tablicy interaktywnej zawartość sekcji „Wprowadzenie" i omawia cele do osiągnięcia w trakcie lekcji.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu, np.
   – co to jest system binarny?
   – podaj algorytm konwersji liczb zapisanych w postaci dziesiętnej do ich dwójkowych odpowiedników,
   – jak konwertujemy część ułamkową z systemu dziesiętnego do postaci binarnej?
   Chętni lub wybrani uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. Odnosząc się do treści zawartej w sekcji „Film samouczek", uczniowie w parach konstruują alternatywny przykład, rozwiązują go i ewentualnie wskazują możliwości jego zastosowania. Rezultaty omawiane są na forum klasy.
2. **Praca z multimedium.** Nauczyciel czyta polecenie nr 1 z sekcji „Przeczytaj". Prosi uczniów, aby wykonali je w parach. Następnie wybrana osoba prezentuje propozycję odpowiedzi, a pozostali uczniowie ustosunkowują się do niej. Nauczyciel w razie potrzeby uzupełnia ją, udziela też uczniom informacji zwrotnej.
3. **Ćwiczenie umiejętności**. Poszukiwanie najefektywniejszego rozwiązania problemu. Uczniowie wykonują w parach polecenie 2 z sekcji „Prezentacja multimedialna", a następnie porównują swój kod omawiając go wspólnie na forum. Nauczyciel ocenia efektywność zastosowanego rozwiązania.
4. Uczniowie w parach wykonują ćwiczenia nr 1-3. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Wybrany uczeń podsumowuje zajęcia z programowania w C++, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie wykonują polecenie nr 2 z sekcji „Prezentacja multimedialna”.

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:

- Nauczyciel może wykorzystać multimedium w sekcji „Film samouczek” do pracy przed lekcją. Uczniowie zapoznają się z jego treścią i przygotowują do pracy na zajęciach w ten sposób, żeby móc samodzielnie rozwiązać zadania dołączone do e-materiału „Konwersja liczb z systemu dziesiętnego na dwójkowy w języku C++”.