

Wydział budowy maszyn i informatyki

Architektura komputerów

(Laboratorium №3)

Temat ćwiczenia: Systemy kodowania liczb

Data wykonania ćwiczenia: 22.11.2023

Igor Gawłowicz

Zadanie 1

Napisz program, który:

- przeliczy podaną przez użytkownika liczbę dziesiętną na jej odpowiednik w systemie binarnym, ósemkowym i szesnastkowym;
- obliczy sumę, różnicę, iloczyn w systemie dwójkowym. Funkcja powinna przyjmować dwa argumenty w postaci ciągów znaków (np. "1010" i "1101") i zwracać wynik również w postaci ciągu znaków;
- wczyta liczbę w systemie ósemkowym od użytkownika, przeliczy ją na system dziesiętny, a następnie wyświetli jej reprezentację binarną;
- który przeliczy podaną przez użytkownika liczbę szesnastkową na jej odpowiednik w systemie dziesiętnym i binarnym.

Zacniemy od przeliczania liczb na odpowiednie typy

Decimal -> Binary

```
std::string decimalToBinary(int num) {
    std::string binary = "";
    if (num == 0) {
        binary = "0";
    } else {
        while (num > 0) {
            binary = (num % 2 == 0 ? "0" : "1") + binary;
            num /= 2;
        }
    }
    return binary;
}
```

Decimal -> Octal

```
std::string decimalToOctal(int num) {
    std::string octal = "";
    if (num == 0) {
        octal = "0";
    } else {
        while (num > 0) {
            // Building the octal representation by appending the remainder of
            division by 8
            octal = std::to_string(num % 8) + octal;
            num /= 8;
        }
    }
    return octal;
}
```

Decimal -> Hexadecimal

```
std::string decimalToHexadecimal(int num) {
    std::string hex = "";
    if (num == 0) {
        hex = "0";
    } else {
        while (num > 0) {
            int remainder = num % 16;
            char digit;
            if (remainder < 10) {
                digit = '0' + remainder;
            } else {
                digit = 'A' + (remainder - 10);
            }
            hex = digit + hex;
            num /= 16;
        }
    }
    return hex;
}
```

Następnie w funkcji main możemy uruchomić nasz kod

```
int main() {
    int number = 555;
    std::string binary = decimalToBinary(number);
    std::string octal = decimalToOctal(number);
    std::string hexadecimal = decimalToHexadecimal(number);

    std::cout << "Binary representation: " << binary << std::endl;
    std::cout << "Octal representation: " << octal << std::endl;
    std::cout << "Hexadecimal representation: " << hexadecimal << std::endl;

    return 0;
}
```

W wyniku dla liczby **555** otrzymamy

```
Binary representation: 1000101011
Octal representation: 1053
Hexadecimal representation: 22B
```

Następnym krokiem będzie napisanie kalkulatora przyjmującego liczby binarne i zwracającego wynik także w liczbach binarnych

Dodawanie

```

std::string binaryAddition(const std::string& bin1, const std::string& bin2) {
    int carry = 0;
    std::string result = "";
    int length = std::max(bin1.length(), bin2.length());

    for (int i = 0; i < length || carry; ++i) {
        int digit1 = (i < bin1.length()) ? bin1[bin1.length() - 1 - i] - '0' : 0;
        int digit2 = (i < bin2.length()) ? bin2[bin2.length() - 1 - i] - '0' : 0;
        int sum = digit1 + digit2 + carry;
        result = char(sum % 2 + '0') + result;
        carry = sum / 2;
    }
    return result;
}

```

Odejmowanie

```

std::string binarySubtraction(const std::string& bin1, const std::string& bin2) {
    std::string result = "";
    int borrow = 0;
    int length = std::max(bin1.length(), bin2.length());

    for (int i = 0; i < length || borrow; ++i) {
        int digit1 = (i < bin1.length()) ? bin1[bin1.length() - 1 - i] - '0' : 0;
        int digit2 = (i < bin2.length()) ? bin2[bin2.length() - 1 - i] - '0' : 0;

        digit1 -= borrow;
        if (digit1 < 0) {
            digit1 += 2;
            borrow = 1;
        } else {
            borrow = 0;
        }

        int diff = digit1 - digit2;
        if (diff < 0) {
            diff += 2;
            borrow = 1;
        }

        result = char(diff % 2 + '0') + result;
    }

    while (result.length() > 1 && result[0] == '0') {
        result = result.substr(1);
    }

    return result;
}

```

Mnożenie

```
std::string binaryMultiplication(const std::string& bin1, const std::string& bin2)
{
    std::string result = "0";
    for (int i = bin2.length() - 1; i >= 0; --i) {
        if (bin2[i] == '1') {
            std::string temp = bin1;
            int padding = bin2.length() - i - 1;
            while (padding-- > 0) {
                temp += "0";
            }
            result = binaryAddition(result, temp);
        }
    }
    return result;
}
```

Następnie możemy sprawdzić wyniki w mainie

```
int main() {
    int number = 555;
    int secondNumber = 777;
    std::string binary = decimalToBinary(number);
    std::string secondBinary = decimalToBinary(secondNumber);

    std::cout << "Binary addtion: " << binaryAddition(binary, secondBinary) <<
    std::endl;

    std::cout << "Binary Subtraction: " << binarySubtraction(secondBinary, binary)
    << std::endl;

    std::cout << "Binary Multiplication: " << binaryMultiplication(binary,
    secondBinary) << std::endl;

    return 0;
}
```

```
Binary addtion: 10100110100
Binary Subtraction: 11011110
Binary Multiplication: 1101001010010000011
```

Następnym punktem jest konwersja z liczby ósemkowej do dziesiętnej a następnie do binarnej

Potrzebujemy do tego funkcji, która konwertuje z liczby oktalnej do dekadonalnej

```
int octalToDecimal(const std::string& octal) {
    int decimal = 0;
    int power = 0;
    for (int i = octal.length() - 1; i >= 0; --i) {
        int digit = octal[i] - '0';
        decimal += digit * pow(8, power);
        ++power;
    }
    return decimal;
}
```

I teraz w mainie wywołamy utworzoną funkcję

```
int main() {
    int number = 555;
    std::string octal = decimalToOctal(number);

    std::cout << "Octal: " << octal << std::endl;

    int decimal = octalToDecimal(octal);

    std::cout << "decimal " << decimal << std::endl;

    std::string binary = decimalToBinary(decimal);

    std::cout << "Binary: " << binary << std::endl;

    return 0;
}
```

Po czym otrzymamy następujące wynik

```
Octal: 1053
decimal 555
Binary: 1000101011
```

Następnym krokiem będzie przemiana liczby szesnastkowej na dziesiętną i binarną

Wykorzystamy do tego funkcję konwertującą z hexa na dziesiętkowy

```
int hexadecimalToDecimal(const std::string& hex) {
    int decimal = 0;
    int power = 0;
    for (int i = hex.length() - 1; i >= 0; --i) {
        int digit;
        if (hex[i] >= '0' && hex[i] <= '9') {
```

```

        digit = hex[i] - '0';
    } else if (hex[i] >= 'A' && hex[i] <= 'F') {
        digit = hex[i] - 'A' + 10;
    } else if (hex[i] >= 'a' && hex[i] <= 'f') {
        digit = hex[i] - 'a' + 10;
    }
    decimal += digit * pow(16, power);
    ++power;
}
return decimal;
}

```

I otrzymamy taki wynik

```

int main() {
    int number = 555;
    std::string hex = decimalToHexadecimal(number);

    std::cout << "Hex: " << hex << std::endl;

    int decimal = hexadecimalToDecimal(hex);

    std::cout << "decimal " << decimal << std::endl;

    std::string binary = decimalToBinary(decimal);

    std::cout << "Binary: " << binary << std::endl;

    return 0;
}

```

```

Hex: 22B
decimal 555
Binary: 1000101011

```

Zadanie 2

Napisz program, który wykonuje operacje:

- dodawania dwóch liczb binarnych;
- odejmowania liczb binarnych;
- mnożenia dwóch liczb binarnych;
- dzielenia liczb binarnych.

Najpierw napisałem funkcję dodawania liczb binarnych

```

std::string binaryAddition(std::string binary1, std::string binary2) {
    std::string result = "";
    int carry = 0;

    int i = binary1.length() - 1;
    int j = binary2.length() - 1;

    while (i >= 0 || j >= 0 || carry == 1) {
        int sum = carry;

        if (i >= 0) {
            sum += binary1[i] - '0';
            i--;
        }

        if (j >= 0) {
            sum += binary2[j] - '0';
            j--;
        }

        result = char(sum % 2 + '0') + result;
        carry = sum / 2;
    }

    return result;
}

```

Następnie odejmowania liczb binarnych

```

std::string binarySubtraction(std::string binary1, std::string binary2) {
    std::string result = "";
    int borrow = 0;

    int i = binary1.length() - 1;
    int j = binary2.length() - 1;

    while (i >= 0 || j >= 0) {
        int num1 = (i >= 0) ? binary1[i--] - '0' : 0;
        int num2 = (j >= 0) ? binary2[j--] - '0' : 0;

        int diff = num1 - num2 - borrow;
        if (diff < 0) {
            diff += 2;
            borrow = 1;
        } else {
            borrow = 0;
        }

        result = char(diff + '0') + result;
    }
}

```



```
    return result;
}
```

Po czym mnożenia liczb binarynych

```
std::string binaryMultiplication(std::string binary1, std::string binary2) {
    int len1 = binary1.length();
    int len2 = binary2.length();
    std::string result(len1 + len2, '0');

    for (int i = len1 - 1; i >= 0; i--) {
        for (int j = len2 - 1; j >= 0; j--) {
            int mul = (binary1[i] - '0') * (binary2[j] - '0');
            int sum = mul + (result[i + j + 1] - '0');

            result[i + j] += sum / 2;
            result[i + j + 1] = char(sum % 2 + '0');
        }
    }

    result.erase(0, std::min(result.find_first_not_of('0'), result.size() - 1));
    return result;
}
```

Niestety miałem spore problem z dzieleniem liczb binarynych i ostatecznie nie udało mi się napisać działającej funkcji.

Jednak dla napisanych podpunktów przygotowałem maina do którego podajemy dwie liczby binarne i otrzymujemy wyniki działań na nich.

```
int main() {
    std::string binaryNum1, binaryNum2;
    std::cout << "Podaj pierwsza liczbe binarna: ";
    std::cin >> binaryNum1;
    std::cout << "Podaj druga liczbe binarna: ";
    std::cin >> binaryNum2;

    std::string additionResult = binaryAddition(binaryNum1, binaryNum2);
    std::cout << "Dodawanie: " << additionResult << std::endl;

    std::string subtractionResult = binarySubtraction(binaryNum1, binaryNum2);
    std::cout << "Odejmowanie: " << subtractionResult << std::endl;

    std::string multiplicationResult = binaryMultiplication(binaryNum1,
    binaryNum2);
    std::cout << "Mnozenie: " << multiplicationResult << std::endl;

    return 0;
}
```



```
Podaj pierwsza liczbe binarna: 1001111
Podaj druga liczbe binarna: 101101
Dodawanie: 1111100
Odejmowanie: 0100010
Mnozenie: 110111100011
```

Zadanie 3

2.3. Napisz program, który wykonuje arytmetyczne/logiczne przesunięcie o n bitów:

- $(x \ll 1) \& (y \gg 1)$
- $((x \wedge y) \gg 2) \mid (y \ll 3)$
- $(\sim x \& (y \ll 2)) \wedge ((x \ll 1) \gg 1)$

zadaniu przyjęto następujące oznaczenie:

- $\ll n$ - arytmetyczne/logiczne przesunięcie o n bitów w lewo;
-  n - logiczne przesunięcie o n bitów w prawo
-  n - arytmetyczne przesunięcie o n bitów w prawo

Tak więc na podstawie podanych przez użytkownika zmiennych x , y oraz n wykonujemy operacje przesunięcia bitowego.

Aby wykonać to zadanie wystarczyło tylko przepisać te operacje w odpowiedni sposób i przedstawić wyniki.

```
int main() {
    int x, y, n;
    std::cout << "Podaj liczbe x: ";
    std::cin >> x;
    std::cout << "Podaj liczbe y: ";
    std::cin >> y;
    std::cout << "Podaj liczbe n: ";
    std::cin >> n;

    int operation1 = (x << 1) & (y >> 1);
    int operation2 = ((x ^ y) >> 2) | (y << 3);
    int operation3 = (~x & (y << 2)) ^ ((x << 1) >> 1);

    std::cout << "Wynik operacji (x << 1) & (y >> 1): " << operation1 <<
std::endl;
    std::cout << "Wynik operacji ((x ^ y) >> 2) | (y << 3): " << operation2 <<
std::endl;
    std::cout << "Wynik operacji (~x & (y << 2)) ^ ((x << 1) >> 1): " <<
operation3 << std::endl;

    return 0;
}
```

Przykładowo testując otrzymałem taki wynik

```
Podaj liczbe x: 5
Podaj liczbe y: 3
Podaj liczbe n: 1
Wynik operacji (x << 1) & (y >> 1): 0
Wynik operacji ((x ^ y) >> 2) | (y << 3): 25
Wynik operacji (~x & (y << 2)) ^ ((x << 1) >> 1): 13
```

Wnioski

Zadanie 1

1. Przeliczanie liczb na różne systemy:

- Program umożliwia przeliczanie liczb dziesiętnych na systemy binarne, ósemkowe i szesnastkowe.
- Wykorzystano funkcje konwertujące liczby dziesiętne na systemy binarne, ósemkowe i szesnastkowe.

2. Operacje arytmetyczne na liczbach binarnych:

- Program wykonuje operacje arytmetyczne na liczbach binarnych, takie jak dodawanie, odejmowanie i mnożenie.
- Napisano funkcje dla każdej operacji, które przyjmują dwie liczby binarne i zwracają wynik również w postaci liczby binarnej.
- Nie zaimplementowano funkcji dzielenia liczb binarnych.

3. Przeliczanie z systemu ósemkowego i szesnastkowego:

- Program przelicza liczby ósemkowe na dziesiętne, a następnie wyświetla ich reprezentację binarną.
- Wykorzystano funkcje konwertujące liczby ósemkowe i szesnastkowe na dziesiętne, a następnie na liczby binarne.

Zadanie 2

1. Operacje na liczbach binarnych:

- Program umożliwia wykonywanie operacji matematycznych na liczbach binarnych, takich jak dodawanie, odejmowanie i mnożenie.
- Napisano funkcje realizujące dodawanie, odejmowanie i mnożenie liczb binarnych.
- Działanie dzielenia liczb binarnych nie zostało zaimplementowane.

Zadanie 3

1. Operacje przesunięcia bitowego:

- Program wykonuje operacje przesunięcia bitowego na podstawie zmiennych x, y i n.

- Przyjmuje oznaczenia $\ll n$ jako przesunięcie bitowe w lewo, $\gg n$ jako przesunięcie bitowe w prawo oraz $\ggg n$ jako arytmetyczne przesunięcie bitowe w prawo.
- Prezentuje wyniki operacji przesunięć bitowych zgodnie z podanymi wzorami matematycznymi.