

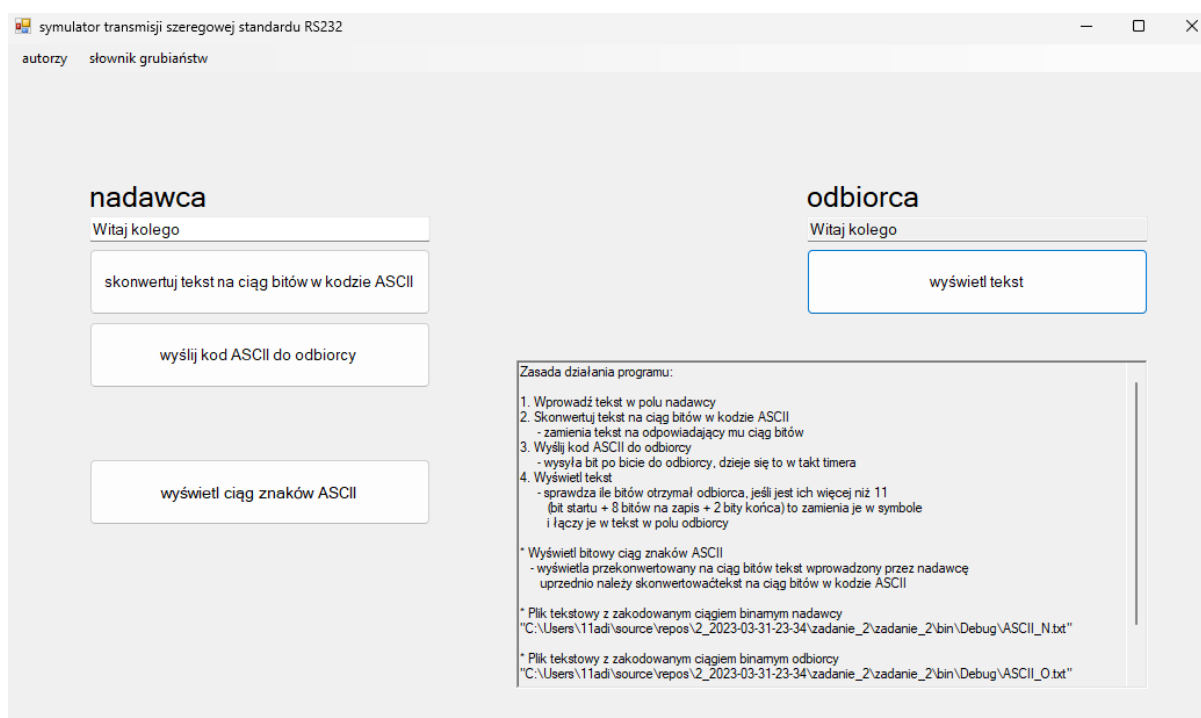
Sprawozdanie zadanie 2

“Symulator transmisji szeregowej standardu RS232”

Organizacja systemów komputerowych

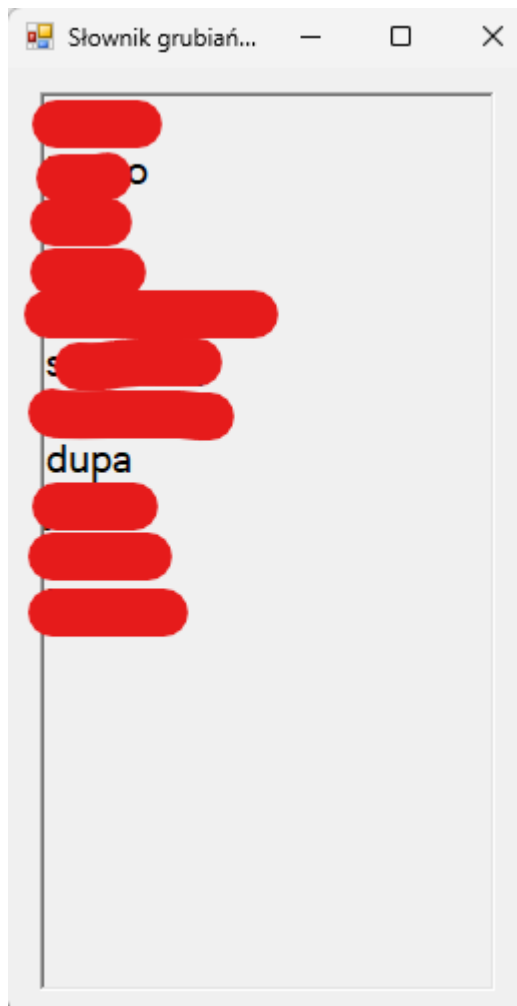
Adrian Nowogrodzki 184332

Oskar Nowak 184289

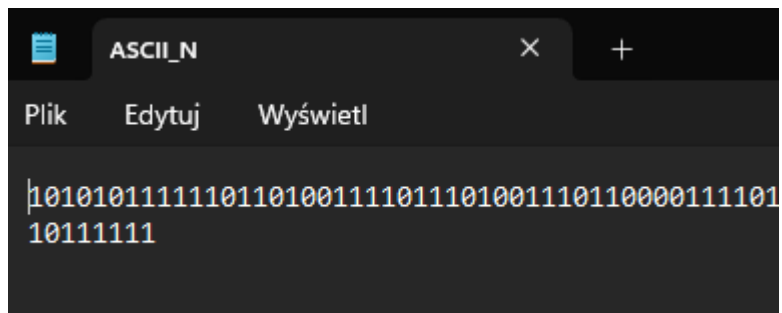


Program umożliwia przesyłanie tekstu wprowadzonego przez nadawcę do innego użytkownika (odbiorcy). Użytkownik wprowadza w polu nadawcy tekst, który po wciśnięciu przycisku konwersji zamieniany jest na ekwiwalentny mu ciąg bitów zapisu ASCII, można ten kod wyświetlić przyciskiem “wyświetl ciąg znaków ASCII”. Tekst po wciśnięciu przycisku konwersji jest również zapisywany do pliku tekstowego w formie odpowiadających mu w zapisie bitowym zer i jedynek wraz z dodatkowymi bitem startu i dwoma bitami końca [bit startu + 8 bitów na symbol + 2 bity końca]. Taki kod można wysłać do odbiorcy przyciskiem wysyłania, po jego wciśnięciu odbywa się transmisja szeregową bitów z pliku “ASCII_N.txt” do “ASCII_O.txt”. Taki przesył odbywa się bit po bicie to za tym idzie każdy kolejny symbol można odczytać dopiero po wpisaniu do pliku odbiorcy wielokrotności 11 bitów (3 bity dodatkowe i 8 bitów znaku). Żeby rozpocząć wyświetlanie tekstu po stronie odbiorcy należy wcisnąć przycisk wyświetlania tekstu po stronie odbiorcy. Następuje wtedy odczyt ciągu bitów z pliku odbiorcy, dekodowanie bitów startu i końca i bieżące tłumaczenie binarnego kodu znaków na tekst. Dbając o kulturę wypowiedzi wprowadzono słownik grubiaństw, które uniemożliwiają wysłanie niecenzuralnych słów do odbiorcy.


Wyświetlanie słownika grubiaństw



Plik tekstowy odbiorcy

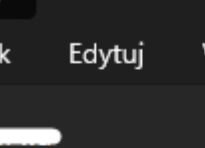


Plik tekstowy nadawcy



1010101111110110100111101110100111011000011110110
10111111

Plik tekstowy słów zakazanych



Grubianstwa

Plik Edytuj Wyświetl

dupa

grubianstwa

Do cenzurowania słów użyto metody Replace z nie uwzględnieniem wielkości liter

```
tekst_nad_s = tekst_nad_s.Replace(line, temp_s, StringComparison.CurrentCultureIgnoreCase);
```

Konwersja z tekstu na ciąg bitów odbywa się poprzez przejście z symboli na ich reprezentację dziesiętną (1 bajt na znak), potem zamianę każdego bajtu na jego ciąg bitowy (BitConverter)

```
tekst_nad_byte = Encoding.ASCII.GetBytes(tekst_nad_s);  
tekst_nad_bitArray = new BitArray(tekst_nad_byte);
```

następnie dodawany jest bit początkowy

```
tekst_nad_bin_s = tekst_nad_bin_s + "1";
```

oraz zapisywane są bity z BitArray do stringa
(np BitArray = 01101100 -> string = "01101100")

```
if (tekst_nad_bitArray[j + 8 * n].ToString() == "False")  
{  
    tekst_nad_bin_s = tekst_nad_bin_s + "0";  
}  
else  
{  
    tekst_nad_bin_s = tekst_nad_bin_s + "1";  
}
```

i dodawane są 2 bity końca

```
tekst_nad_bin_s = tekst_nad_bin_s + "11";
```

Wyświetlanie bitów działa analogicznie, z taką różnicą, że ciąg bitów pozbawiony jest bitu startu i bitów końca.

Przesyłanie danych do odbiorcy zachodzi bit po bicie w takt timera

```
private void timer1_Tick(object sender, EventArgs e)
{
    // PRZEPISYWANIE KOLEJNYCH PO SOBIE BITÓW DANYCH W PLIKU NADAWCY Z PLIKU NADAWCY DO ODBIORCY
    przepisywanie_kolejnych_po_sobie_bitow_danych_w_pliku_nadawcy_z_pliku_nadawcy_do_odbiorcy();

    // Wyświetlanie tekstu w oknie odbiorcy
    if (wyswietl_0 == 1)
    {
        wyswietlanie_tekstu_w_oknie_odbiorcy();
    }
}
```

Przesył zrealizowany jest następująco:

Czytnik ustawia się na wskazanym miejscu

```
fs.Position = pozycja_czytnika_i;
```

Odczytywany jest jeden znak (0/1), ale zapisany w znak_odbyte jako odpowiadający 0/1 (byte)ASCII, czyli (char)0->(byte/int)48 | 1->49)

```
fs.Read(znak_odbyte, pozycja_czytnika_i, 1);
```

Integer/byte ten jest zamieniany na odpowiedni znak (char)

```
znak_odc = Convert.ToChar(znak_odbyte[pozycja_czytnika_i]);
```

odczytany znak przepisywany jest do pliku odbiorcy

```
sb.Position = pozycja_czytnika_i;
sw.Write(znak_odc);
```

a pozycja czytnika zmienia się

```
pozycja_czytnika_i++;
```

Odczyt danych z pliku odbiorcy oraz wyświetlanie:

Czytany jest dotychczasowo wypełniony plik odbiorcy

```
tekst_odb_s0 = File.ReadAllText("ASCII_0.txt");
```

Jeśli ilość bitów w pliku tekstowym jest wielokrotnością 11 to oznacza, że można odczytać/wyodrębnić z kodu nowy znak.

```
while ((tekst_odb_s0.Length - 11 * ilosc_znakow_odb_i) >= 11)
{
    tekst_odb_s1[ilosc_znakow_odb_i] = tekst_odb_s0.Substring(1 + 11 * ilosc_znakow_odb_i, 8);
    ilosc_znakow_odb_i++;
}
```

Następuje zamiana ciągu bitów zapisanych w stringu do odpowiadającego mu tekstu oraz następuje jego wyświetlenie w oknie odbiorcy.

```
// ZAMIANA CIĄGU BITÓW W SYMBOLE I ŁĄCZENIE ICH W TEKST ODBIORCY
tekst_odb_s2 = string.Join("", tekst_odb_s1);
tekst_odb_s = "";
while (tekst_odb_s2.Length > 0)
{
    first8 = tekst_odb_s2.Substring(0, 8);
    tekst_odb_s2 = tekst_odb_s2.Substring(8);
    number = Convert.ToInt32(first8, 2);
    tekst_odb_s += (char)number;
}
textBox_0.Text = tekst_odb_s;    // wyświetlanie tekstu
```

Co ważne przesył i odczyt uruchamiają się przy pikach timera co ma za zadanie symulować przesył bitów pomiędzy nadawcą a odbiorcą.