```
#include <stdio.h>
#include <stdlib.h>
#include "operacje_proste.h"
#include "konwersje.h"
#include "tokeny.h"


enum Result { OK, ERROR };

void Testof_CopyString()
{
  printf("CopyString\n\n");

  printf("Test 1 - ");
  //ta sama dlugosc wyrazow
  char cTestSource_1[] = "test1";
  char cTestDestination_1[] = "aaaaa";
  CopyString(cTestSource_1, cTestDestination_1);
  if (eCompareString(cTestSource_1, cTestDestination_1) == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //pierwszy wyraz krotszy
  char cTestSource_2[] = "test2";
  char cTestDestination_2[] = "aaaaaaa";
  CopyString(cTestSource_2, cTestDestination_2);
  if (eCompareString(cTestSource_2, cTestDestination_2) == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //Source jest pusty
  char cTestSource_3[] = "";
  char cTestDestination_3[] = "aaaaa";
  CopyString(cTestSource_3, cTestDestination_3);
  if (eCompareString(cTestSource_3, cTestDestination_3) == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 4 - ");
  //Destination jest pusty
  char cTestSource_4[] = "test4";
  char cTestDestination_4[] = "";
  CopyString(cTestSource_4, cTestDestination_4);
  if (eCompareString(cTestSource_4, cTestDestination_4) == DIFFERENT) printf("OK\n"); else printf("Error\n");
}
```

```
void TestOf_eCompareString()
{
  printf("eCompareString\n\n");

  printf("Test 1 - ");
  //takie same, ta sama dlugosc
  if (eCompareString("test1", "test1") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //rozne wyrazy, ta sama dlugosc
  if (eCompareString("test1", "test2") == DIFFERENT) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //jeden wyraz pusty
  if (eCompareString("", "test2") == DIFFERENT) printf("OK\n"); else printf("Error\n");

  printf("Test 5 - ");
  ///oba puste
  if (eCompareString("", "") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 6 - ");
  ///drugi wyraz dluzszy
  if (eCompareString("test1", "test2222") == DIFFERENT) printf("OK\n"); else printf("Error\n");
}

void TestOf_AppendString()
{
  printf("AppendString\n\n");

  printf("Test 1 - ");
  //dowolne
  char cTestSource_1[] = "Pierwszy";
  char cTestDestination_1[] = "Test";
  AppendString(cTestSource_1, cTestDestination_1);
  if (eCompareString(cTestDestination_1, "TestPierwszy") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //source pusty
  char cTestSource_2[] = "";
  char cTestDestination_2[] = "test";
  AppendString(cTestSource_2, cTestDestination_2);
  if (eCompareString(cTestDestination_2, "test") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //destination pusty
  char cTestSource_3[] = "test";
```

```
  char cTestDestination_3[] = "";
  AppendString(cTestSource_3, cTestDestination_3);
  if (eCompareString(cTestDestination_3, "test") == EQUAL) printf("OK\n"); else printf("Error\n");
}

void TestOf_ReplaceCharactersInString()
{
  printf("ReplaceCharactersInString\n\n");

  printf("Test 1 - ");
  //zamiana roznych znakow
  char cTestString_1[] = "test1";
  ReplaceCharactersInString(cTestString_1, '1', '55');
  if (eCompareString(cTestString_1, "test55") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //spacja na NULL
  char cTestString_2[] = "lancuch znakowy";
  ReplaceCharactersInString(cTestString_2, ' ', '\0');
  if (eCompareString(cTestString_2, "lancuch\0znakowy") == EQUAL) printf("OK\n"); else printf("Error\n");
}

void TestOf_UIntToHexStr()
{
  printf("UIntToHexStr\n\n");

  printf("Test 1 - ");
  //zamiana zwyklej liczby
  char cTestDestination[7];

  UIntToHexStr(123, cTestDestination);
  if (eCompareString(cTestDestination, "0x007B") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //krance przedzialow, 0,9,A,F
  UIntToHexStr(2479, cTestDestination);
  if (eCompareString(cTestDestination, "0x09AF") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //czy na koncu jest NULL
  UIntToHexStr(123, cTestDestination);
  if ((eCompareString(cTestDestination, "0x007B") == EQUAL) && (cTestDestination[7] == '\0')) printf("OK\n"); else printf("Error\
n");
}
```

```
void TestOf_eHexStringToUInt()
{
  printf("eHexStringToUInt\n\n");

  printf("Test 1 - ");
  //krance przedzialow 0, 9, A, F
  enum Result eReturnResult;
  unsigned int uiTestDestination;

  eReturnResult = eHexStringToUInt("0x09AF", &uiTestDestination);
  if ((eReturnResult == OK) && (uiTestDestination == 2479)) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //za krotki
  eReturnResult = eHexStringToUInt("0x2D", &uiTestDestination);
  if ((eReturnResult == OK) && (uiTestDestination == 45)) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //za dlugi
  eReturnResult = eHexStringToUInt("0x7C03C", &uiTestDestination);
  if (eReturnResult == ERROR) printf("OK\n"); else printf("Error\n");

  printf("Test 4 - ");
  //brak 0x na poczatku
  eReturnResult = eHexStringToUInt("A8F4", &uiTestDestination);
  if (eReturnResult == ERROR) printf("OK\n"); else printf("Error\n");

  printf("Test 4 - ");
  //sam przedrostek 0x a tak to pusty string
  eReturnResult = eHexStringToUInt("0x", &uiTestDestination);
  if (eReturnResult == ERROR) printf("OK\n"); else printf("Error\n");
}

void TestOf_AppendUIntToString()
{
  printf("AppendUIntToString\n\n");

  printf("Test 1 - ");
  //niepusty string
  char cTestString_1[] = "TestString";
  AppendUIntToString(60, cTestString_1);
  if (eCompareString(cTestString_1, "TestString0x003C") == EQUAL) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //pusty string
  char pcTestString_2[] = "";
```

```
  AppendUIntToString(60, cTestString_2);
  if (eCompareString(cTestString_2, "0x003C") == EQUAL) printf("OK\n"); else printf("Error\n");
}

void TestOf_ucFindTokensInString()
{
  unsigned char ucTokenNumber;

  printf("ucFindTokensInString\n\n");

  printf("Test 1 - ");
  //max liczba tokenów
  char cTestString_1[] = "Ola ma jeża";
  ucTokenNumber = ucFindTokensInString(cTestString_1);
  if ((ucTokenNumber == 3)&&(&cTestString_1[0] == asToken[0].uValue.pcString)&&(&cTestString_1[4] ==
asToken[1].uValue.pcString)&&(&cTestString_1[7] == asToken[2].uValue.pcString)) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //tylko delimitery
  char cTestString_2[] = "   ";
  ucTokenNumber = ucFindTokensInString(cTestString_2);
  if (ucTokenNumber == 0) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //delimiter na poczatku stringa
  char cTestString_3[] = " Ola ma jeza";
  ucTokenNumber = ucFindTokensInString(pcTestString_3);
  if ((ucTokenNumber == 3)&&(&cTestString_3[1] == asToken[0].uValue.pcString)&&(&cTestString_3[5] ==
asToken[1].uValue.pcString)&&(&cTestString_3[8] == asToken[2].uValue.pcString)) printf("OK\n"); else printf("Error\n");

  printf("Test 4 - ");
  //wiecej niz 1 delimiter pomiedzy tokenami
  char cTestString_4[] = "Ola  ma  jeza";
  ucTokenNumber = ucFindTokensInString(pcTestString_3);
  if ((ucTokenNumber == 3)&&(&cTestString_4[0] == asToken[0].uValue.pcString)&&(&cTestString_4[5] ==
asToken[1].uValue.pcString)&&(&cTestString_4[9] == asToken[2].uValue.pcString)) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //mniej niz 3 tokeny
  char cTestString_5[] = "Ola ma";
  ucTokenNumber = ucFindTokensInString(pcTestString_3);
  if ((ucTokenNumber == 3)&&(&cTestString_5[0] == asToken[0].uValue.pcString)&&(&cTestString_5[4] == asToken[1].uValue.pcString))
printf("OK\n"); else printf("Error\n");
```

```
  printf("Test 4 - ");
  //za duzo tokenow
  char cTestString_6[] = "Ola ma jeza i psa";
  ucTokenNumber = ucFindTokensInString(pcTestString4);
  if ((ucTokenNumber == 3)&&(&cTestString_6[0] == asToken[0].uValue.pcString)&&(&cTestString_6[4] ==
asToken[1].uValue.pcString)&&(&cTestString_6[7] == asToken[2].uValue.pcString)) printf("OK\n"); else printf("Error\n");
}

void TestOf_eStringToKeyword()
{
  enum KeywordCode eTokenCode;

  printf("eStringToKeyword\n\n");

  printf("Test 1 - ");
  //slowo kluczowe load
  if ((eStringToKeyword("load", &eTokenCode) == OK)&&(eTokenCode == LD)) printf("OK\n"); else printf("Error\n");

  printf("Test 2 - ");
  //slowo kluczowe reset
  if ((eStringToKeyword("reset", &eTokenCode) == OK)&&(eTokenCode == RST)) printf("OK\n"); else printf("Error\n");

  printf("Test 3 - ");
  //slowo kluczowe store
  if ((eStringToKeyword("store", &eTokenCode) == OK)&&(eTokenCode == ST)) printf("OK\n"); else printf("Error\n");

  printf("Test 4 - ");
  //brak slowa kluczowego
  if (eStringToKeyword("token1", &eTokenCode) == ERROR) printf("OK\n"); else printf("Error\n");
}

void TestOf_DecodeTokens()
{
  unsigned char ucTokenNumber;
  char cTestToken_1[] = "load";
  char cTestToken_2[] = "0x20";
  char cTestToken_3[] = "immediately";
  asToken[0].uValue.pcString = &cTestToken_1[0];
  asToken[1].uValue.pcString = &cTestToken_2[0];
  asToken[2].uValue.pcString = &cTestToken_3[0];
  ucTokenNumber = 3;

  printf("DecodeTokens\n\n");
```

```
  printf("Test 1 - ");
  //dekodowanie tokenow
  DecodeTokens();
  if ((asToken[0].eType == KEYWORD)&&(asToken[0].uValue.eKeyword == LD)&&(asToken[1].eType ==
NUMBER)&&(asToken[1].uValue.uiNumber == 32)&&(asToken[2].eType == STRING)&&(asToken[2].uValue.pcString == &cTestToken_3))
printf("OK\n"); else printf("Error\n");
}

void TestOf_DecodeMsg()
{
  char cTestMessage[] = "load 0x20 immediately";

  printf("DecodeMsg\n\n");

  printf("Test 1 - ");
  //dekodowanie calego lancucha
  DecodeMsg(pcTestMsg);
  if ((ucTokenNr == 3)&&(asToken[0].eType == KEYWORD)&&(asToken[0].uValue.eKeyword == LD)&&(asToken[1].eType ==
NUMBER)&&(asToken[1].uValue.uiNumber == 32)&&(asToken[2].eType == STRING)&&(asToken[2].uValue.pcString == &pcTestMsg[10]))
printf("OK\n"); else printf("Error\n");
}

int main()
{
  Testof_CopyString();
    TestOf_eCompareString();
    TestOf_AppendString();
    TestOf_ReplaceCharactersInString();
    TestOf_UIntToHexStr();
    TestOf_eHexStringToUInt();
    TestOf_AppendUIntToString();
    TestOf_ucFindTokensInString();
    TestOf_eStringToKeyword();
    TestOf_DecodeTokens();
    TestOf_DecodeMsg();
}
```