Wstęp
do programowania

SPRAWOZDANIE

Laboratorium nr 4

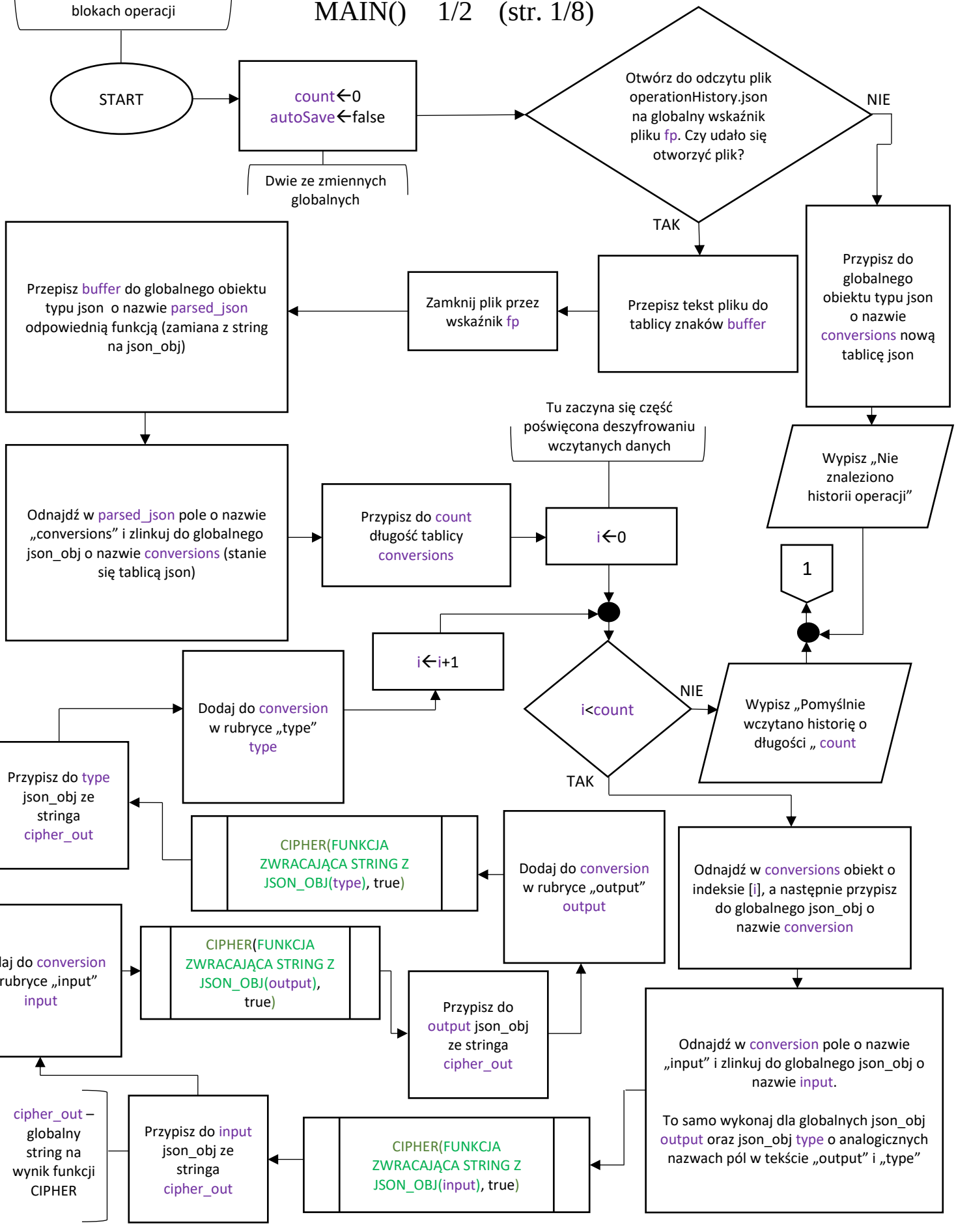
dr inż. Hubert Ostap

Dawid Leszczyński
16.06.2020
WCY19XQ5S1

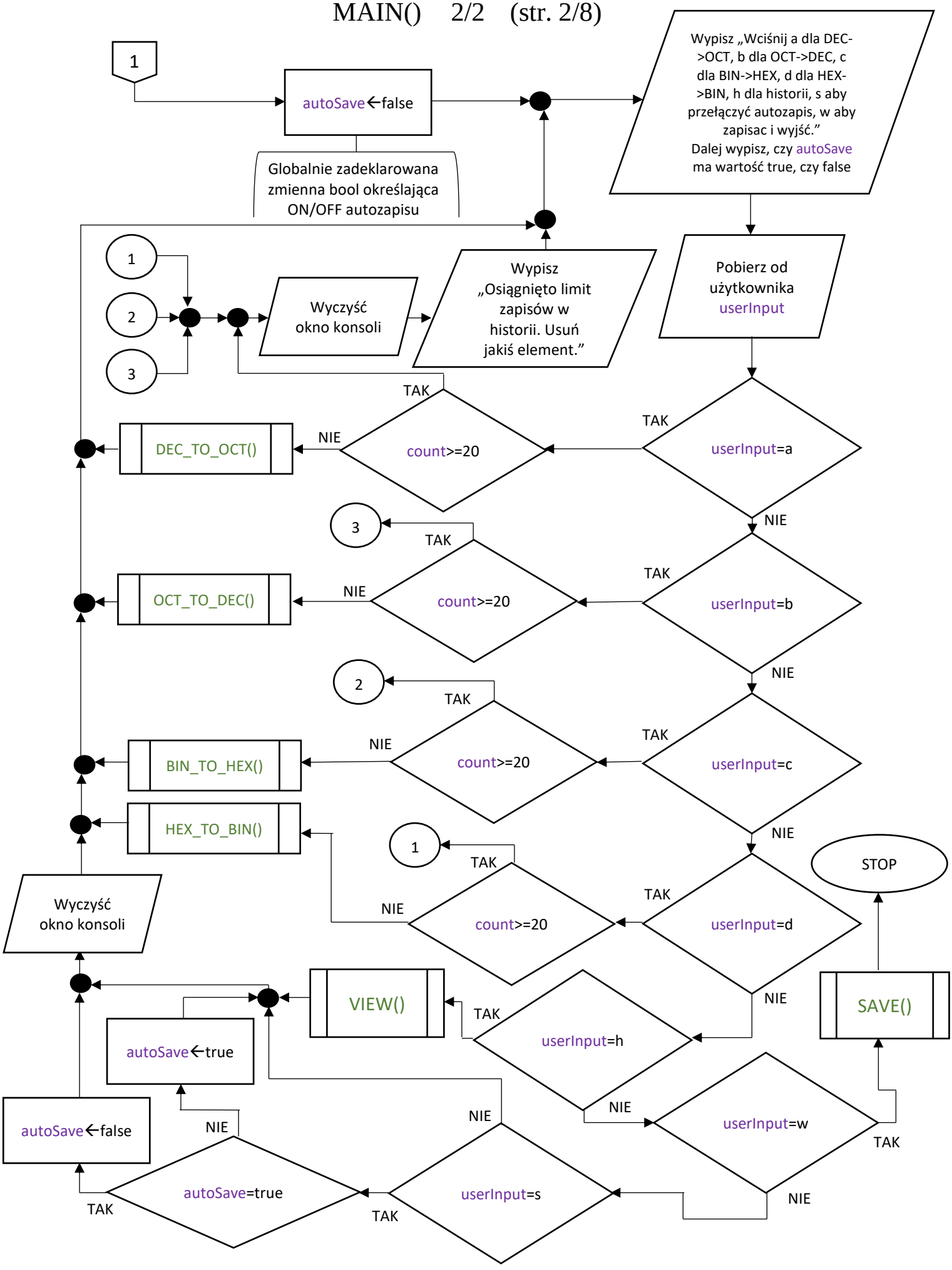
Dla wzmocnionej czytelności kolorem fioletowym oznaczyłem zmienne, czerwonym znaki specjalne przy wypisywaniu tekstu, a zielonym i ciemnym zielonym wywołania funkcji przez nazwę. Funkcje biblioteki JSON opisałem tekstem w blokach operacji

Schemat blokowy

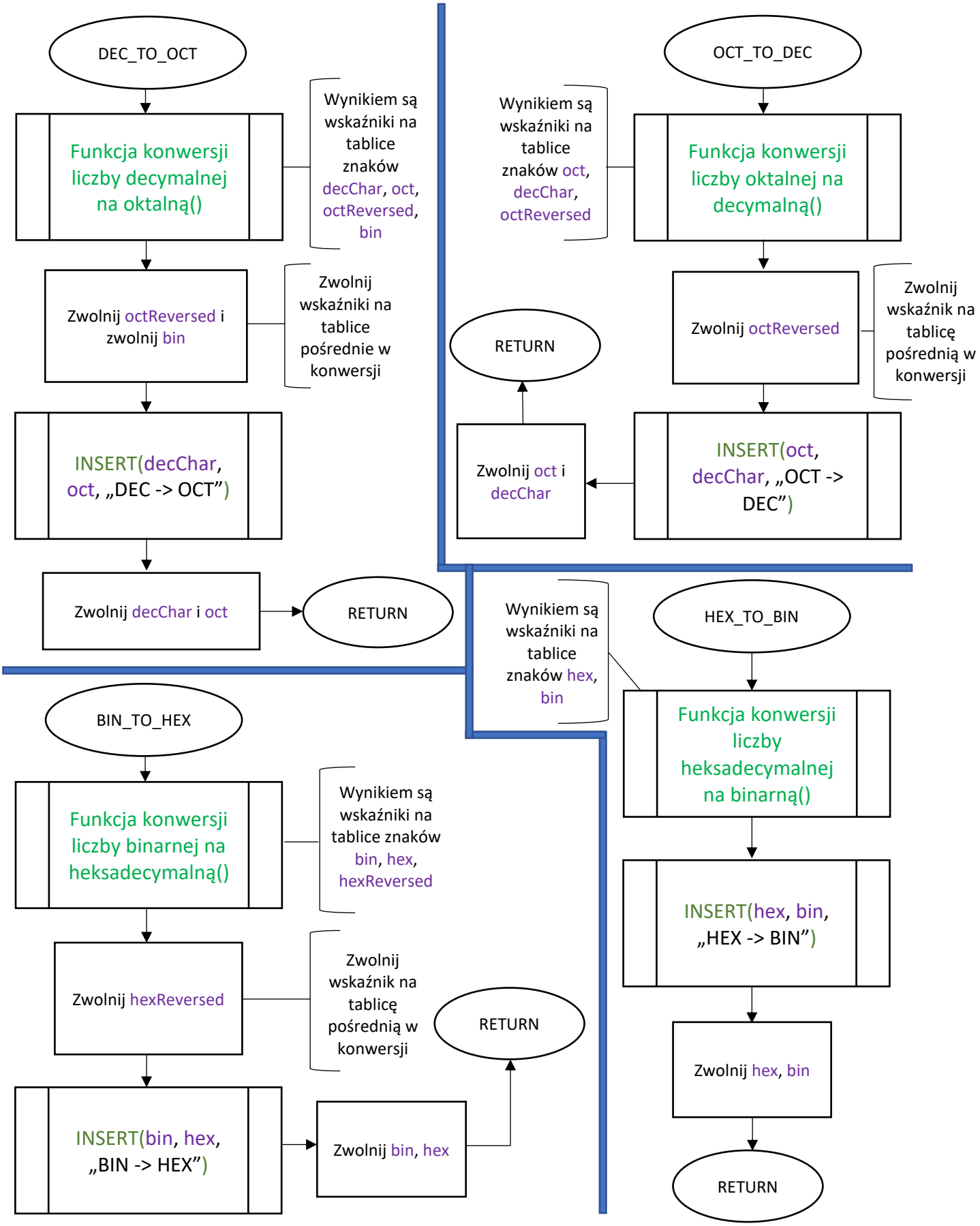
MAIN() 1/2 (str. 1/8)



MAIN() 2/2 (str. 2/8)



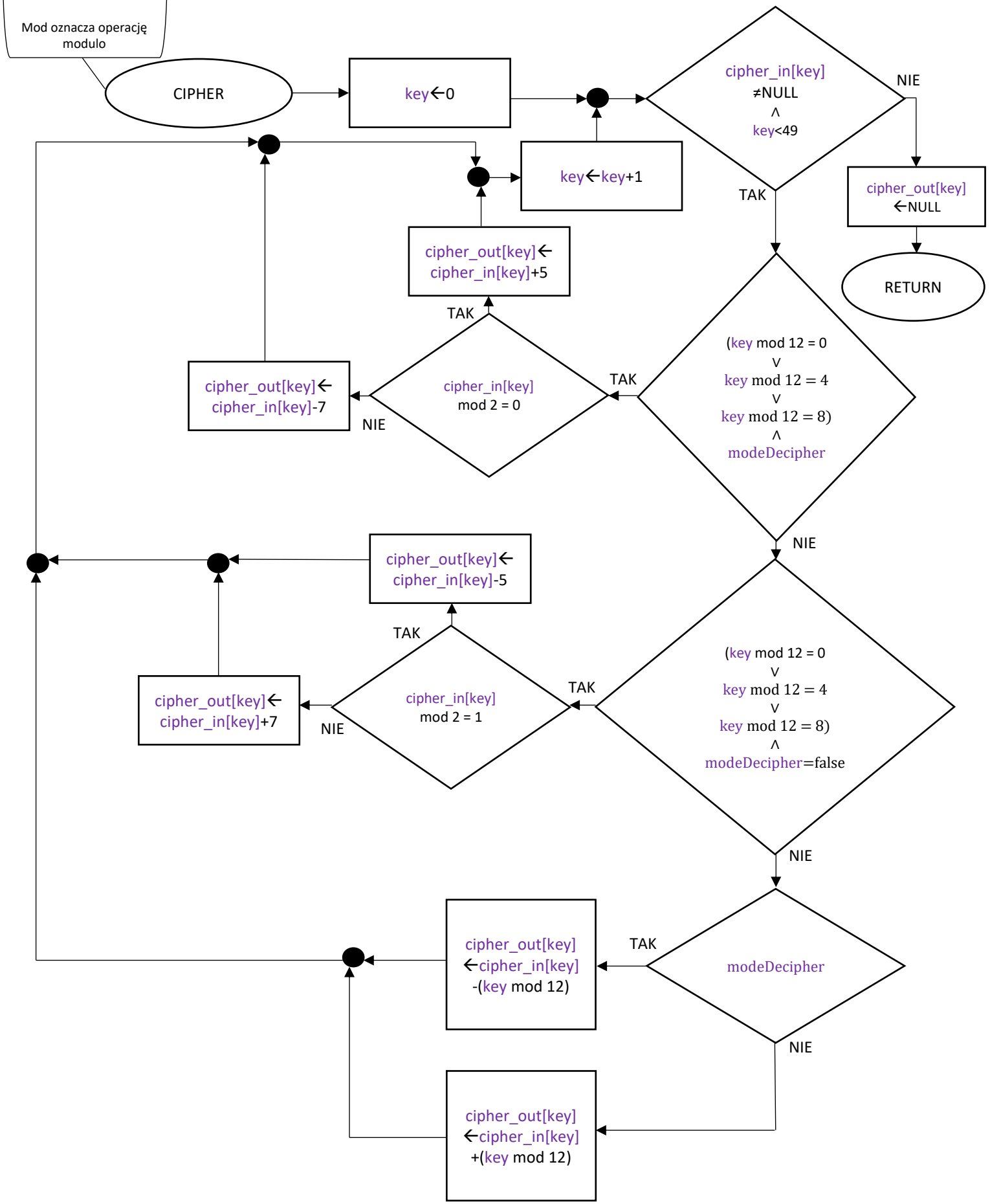
DEC_TO_OCT() OCT_TO_DEC() BIN_TO_HEX() HEX_TO_BIN() (str. 3/8)



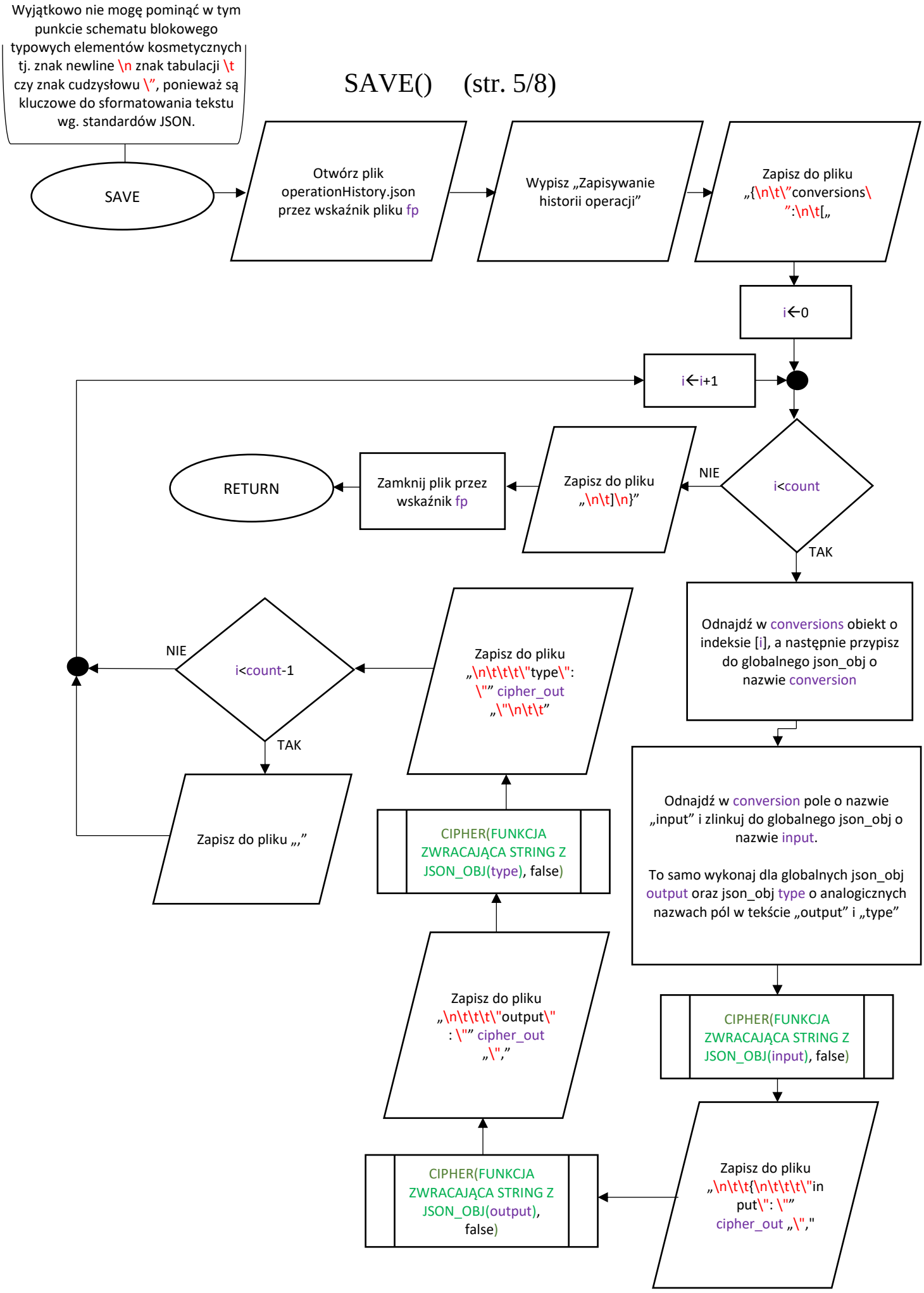
W szyfrowaniu będę porównywał znaki z liczbami i wykonywał na nich działania. Należy to interpretować jako działania na liczbach z tablicy ASCII odpowiednich dla tych znaków

Mod oznacza operację modulo

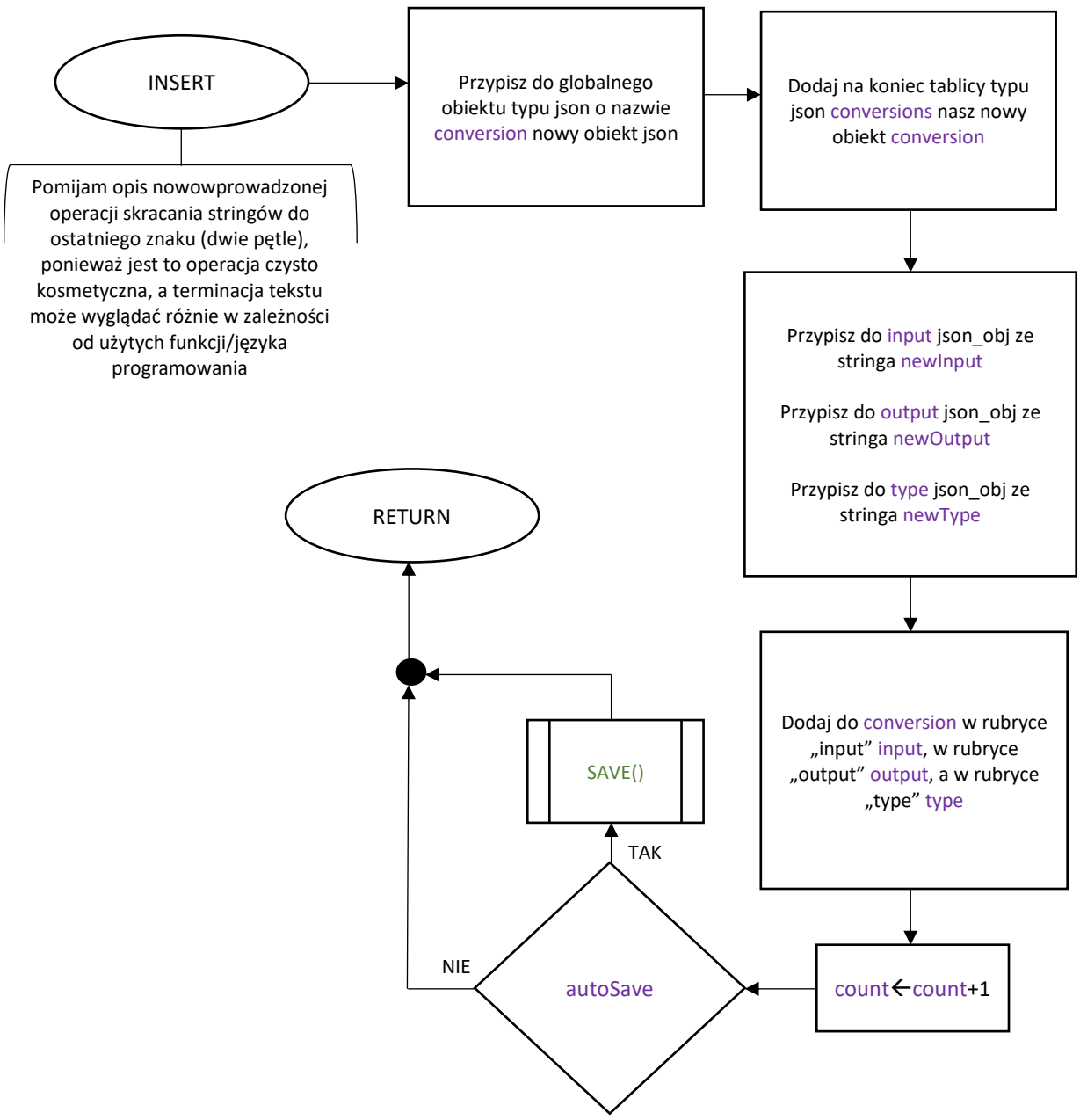
CIPHER(char cipher_in[], bool modeDecipher) (str. 4/8)



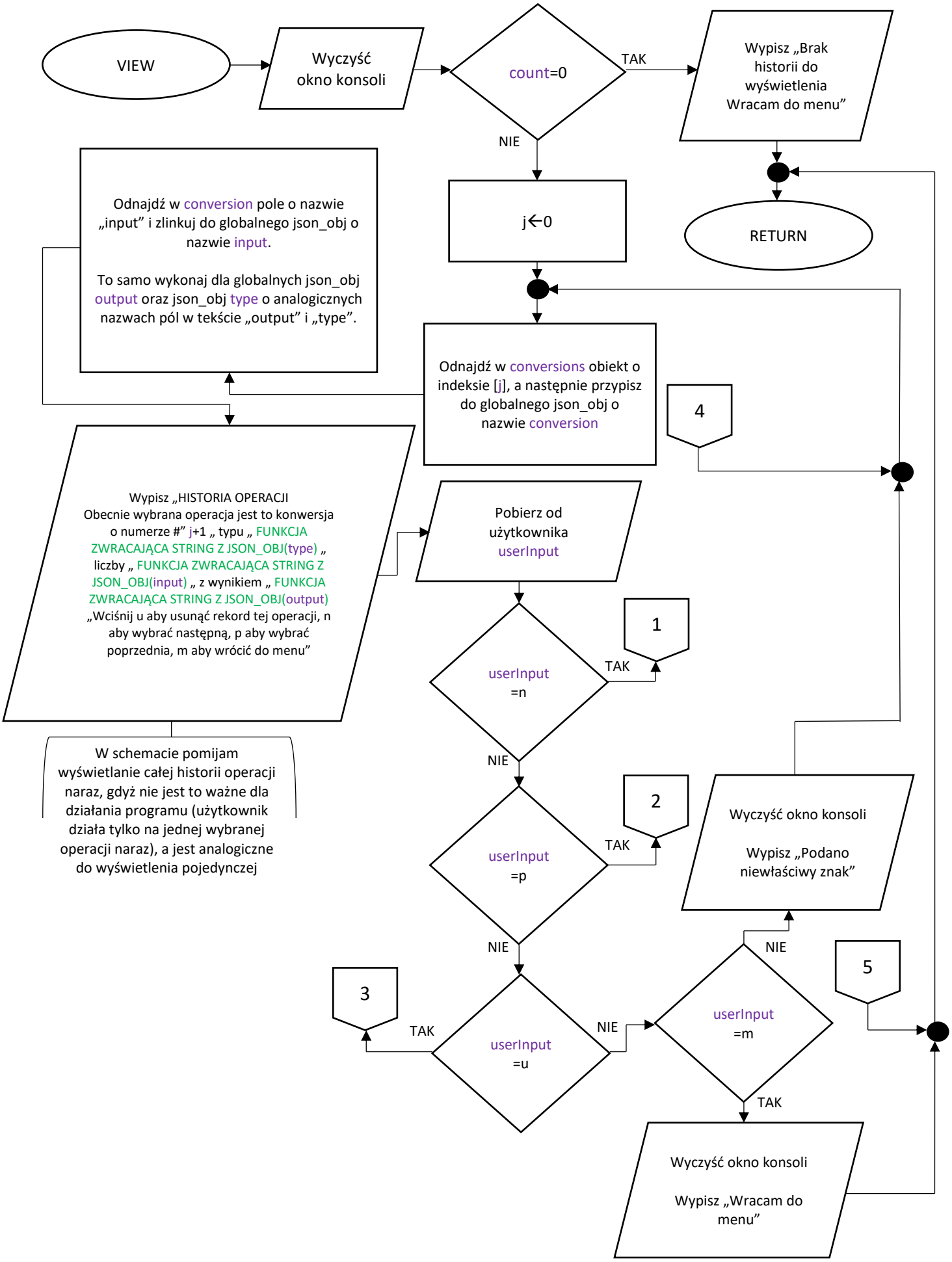
SAVE() (str. 5/8)

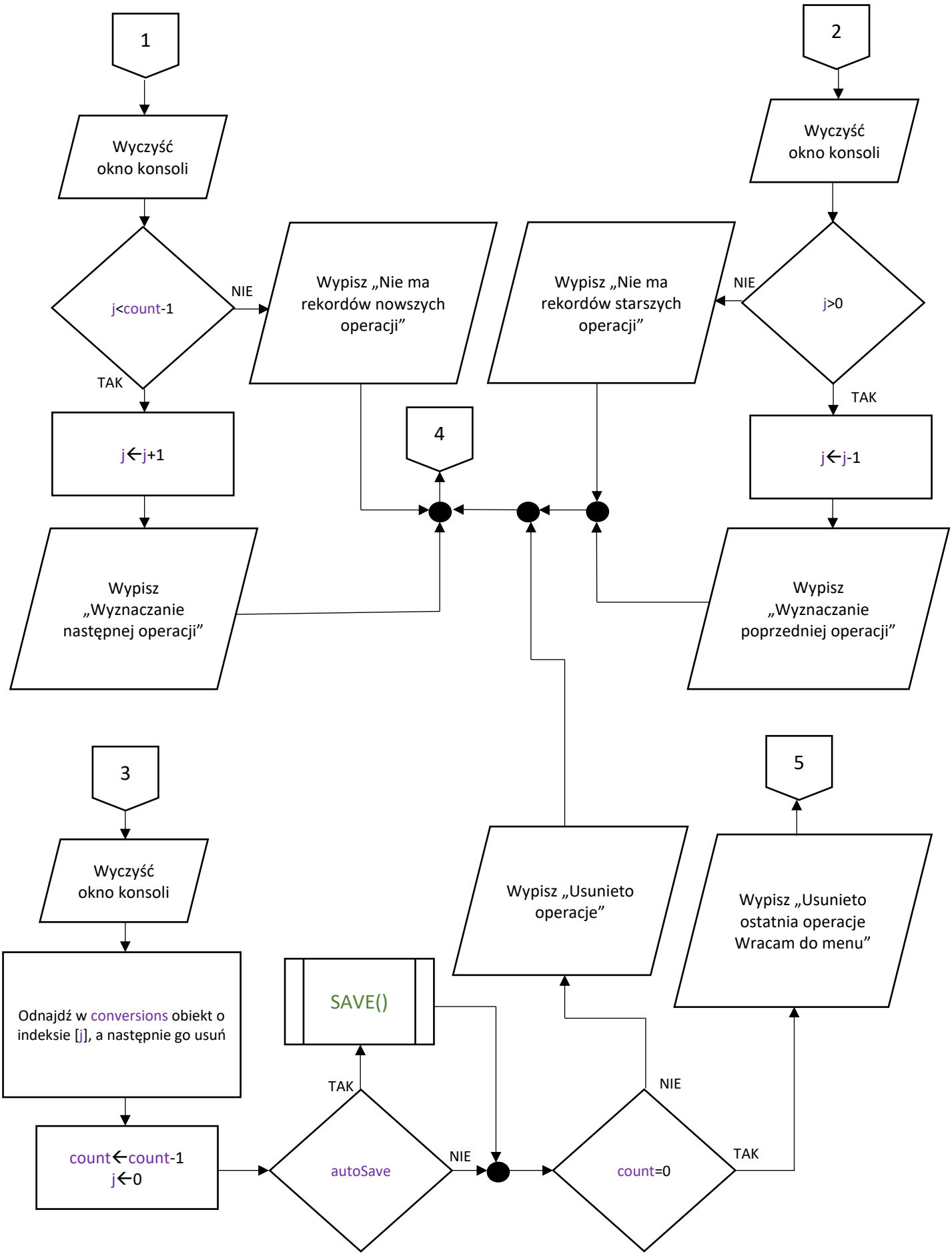

Wyjątkowo nie mogę pominąć w tym punkcie schematu blokowego typowych elementów kosmetycznych tj. znak newline \n znak tabulacji \t czy znak cudzysłowu \", ponieważ są kluczowe do sformatowania tekstu wg. standardów JSON.
SAVE
Otwórz plik operationHistory.json przez wskaźnik pliku fp
Wypisz „Zapisywanie historii operacji”
Zapisz do pliku „{\n\t\"conversions\":\n\t[„
i←0
i←i+1
i<count
NIE
TAK
Zapisz do pliku „\n\t]\n}”
Zamknij plik przez wskaźnik fp
RETURN
Odnajdź w conversions obiekt o indeksie [i], a następnie przypisz do globalnego json_obj o nazwie conversion
Odnajdź w conversion pole o nazwie „input” i zlinkuj do globalnego json_obj o nazwie input.
To samo wykonaj dla globalnych json_obj output oraz json_obj type o analogicznych nazwach pól w tekście „output” i „type”
CIPHER(FUNKCJA ZWRACAJĄCA STRING Z JSON_OBJ(input), false)
Zapisz do pliku „\n\t\t{\n\t\t\t\"input\": \"” cipher_out „\",”
CIPHER(FUNKCJA ZWRACAJĄCA STRING Z JSON_OBJ(output), false)
Zapisz do pliku „\n\t\t\t\"output\" : \"” cipher_out „\",”
CIPHER(FUNKCJA ZWRACAJĄCA STRING Z JSON_OBJ(type), false)
Zapisz do pliku „\n\t\t\t\"type\": \"” cipher_out „\"\n\t\t”
i<count-1
NIE
TAK
Zapisz do pliku „,”

INSERT(char newInput[], char newOutput[], char newType[]) (str. 6/8)



VIEW() (1/2) (str. 7/8)




1
Wyczyść okno konsoli
j<count-1
NIE
Wypisz „Nie ma rekordów nowszych operacji”
TAK
j←j+1
Wypisz „Wyznaczanie następnej operacji”
2
Wyczyść okno konsoli
j>0
NIE
Wypisz „Nie ma rekordów starszych operacji”
TAK
j←j-1
Wypisz „Wyznaczanie poprzedniej operacji”
4
3
Wyczyść okno konsoli
Odnajdź w conversions obiekt o indeksie [j], a następnie go usuń
count←count-1
j←0
autoSave
TAK
SAVE()
NIE
count=0
NIE
Wypisz „Usunieto operacje”
TAK
5
Wypisz „Usunieto ostatnia operacje Wracam do menu”

Kod źródłowy
(Windows, Visual Studio)

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include <json.h>


/*Program umozliwia konwersje liczb z DEC->OCT, OCT->DEC, BIN->HEX, HEX->BIN
Wyniki podaje na ekranie i zapisuje do pliku operationHistory.json,
moze przechowywac max 20 operacji, w przeciwnym razie program nie pozwoli wykonywac wiecej
instrukcji,

Musi byc mozliwa konwersja liczby 75082 (5 znakow) dziesietnej, 7FFE DAB5 (8 znakow)
szesnastkowej,
222 512 (6 znakow) osemkowej, 0111 1111 1111 1110 1101 1010 1011 0101 (31 znakow) binarnej
W tym przypadku pozwolilem sobie na uproszczenie i trzymam sie tylko dlugosci liczb*/

//UWAGA 1 - do czyszczenia okna konsoli wykorzystana jest funkcja systemu Windows, wiec na takim
systemie nalezy programu uzywac

//UWAGA 2 - podanie do konwersji DEC->OCT lub OCT->DEC czegos innego niz cyfry i enter zaskutkuje
crashem, poniewaz program wczytuje tam do zmiennej integer

//GLOBALS
size_t count = 0; //@param {size_t} count - globalna liczba zapisanych operacji w pamieci
struct json_object* parsed_json, * conversions, * conversion, * input, * output, * type; //@param
{json_object*} parsed_json, conversions, conversion, input, output, type - kolejno globalnie
zapisany tekst (caly) pobrany z pliku, tablica konwersji, obiekt konwersja, tekst wejscia, tekst
wyjscia, tekst typu operacji
FILE* fp; //@param {FILE*} fp - globalny wskaznik do pliku historii operacji
bool autoSave = false; //@param {bool} autoSave - globalny modyfikator dla funkcji insert i view,
decyduje o tym, czy po dodawaniu i usuwaniu z historii od razu zapisywac ja do pliku
char cipher_out[50]; //@param {string} cipher_out - globalny bufor dla tekstu wychodzacego z
funkcji cipher


//FUNCTION DECLARATIONS
void view(); //wyswietla historie operacji i pozwala na usuwanie z niej elementow
void insert(char newInput[], char newOutput[], char newType[]); //insert dodaje do tablicy typu
JSON nowy element o parametrach @param {string} newInput, newOutput, newType - kolejno tekst
wejsciowy operacji, wyjsciowy i jej typ
void save(); //zapisuje stan historii operacji w formacie JSON do pliku operationHistory.json,
nadpisujac caly obecny tekst
void cipher(char cipher_in[], bool modeDecipher); //funkcja cipher dekoduje lub koduje (zaleznie
od @param {bool} modeDecipher) tekst @param {string} coded, wynikiem jest aktualizacja globalnej
tablicy znakow @param {string} cipher_out
void dec_to_oct(); //konwersja z decymalnego na oktalny
void oct_to_dec(); //konwersja z oktalnego na decymalny
void bin_to_hex(); //konwersja z binarnego na heksadecymalny
void hex_to_bin(); //konwersja z heksadecymalnego na binarny


//MAIN
int main()
{
   char buffer[1840]; //@param {string} buffer - bufor znakow dla tekstu wczytanego z pliku
operationHistory.json - rozmiar 1840 pozwala na 20 operacji po 92 znaki
```

```
   char userInput; //@param {char} userInput - pobrany od uzytkownika znak
    int i; //@param {int} i - licznik do roznych iteracji

   if (fopen_s(&fp, "operationHistory.json", "r") == 0) //jeżeli pomyślnie wczytano historię operacji, przypisz tekst do bufora i wypisz komunikat
   {
          fread(buffer, 1840, 1, fp); //buffer <- plik
          fclose(fp);

          parsed_json = json_tokener_parse(buffer); //parsed_json <- buffer
          json_object_object_get_ex(parsed_json, "conversions", &conversions); //przypisz tablice nazwana "conversions" z parsed_json do conversions
          count = json_object_array_length(conversions); //ustaw inicjalna wartosc licznika konwersji

          //deszyfrowanie wczytanych danych
          for (i = 0; i < count; i++)
          {
                 conversion = json_object_array_get_idx(conversions, i); //konwersja jako element konwersji o indeksie j

                 json_object_object_get_ex(conversion, "input", &input); //pobierz input z konwersji i przypisz do input
                 cipher(json_object_get_string(input), true); //deszyfruj input
                 input = json_object_new_string(cipher_out); //input jako nowy obiekt JSON (AKA Zbigniew) ze string cipher_out
                 json_object_object_add(conversion, "input", input); //dodaj do konwersji input w odpowiedniej rubryce (nadpisuje stara wartosc, wtedy przy nastepnym wywolaniu i-tej konwersji dostaniemy od razu zdeszyfrowany wynik)

                 json_object_object_get_ex(conversion, "output", &output); //jak wyzej, ale dla output
                 cipher(json_object_get_string(output), true);
                 output = json_object_new_string(cipher_out);
                 json_object_object_add(conversion, "output", output);

                 json_object_object_get_ex(conversion, "type", &type); //jak wyzej, ale dla type
                 cipher(json_object_get_string(type), true);
                 type = json_object_new_string(cipher_out);
                 json_object_object_add(conversion, "type", type);
          }

          printf("Pomyslnie wczytano historie operacji\nZawiera ona nastepujaca liczbe zapisow: %lu\n\n", count);
   }
   else
   {
          conversions = json_object_new_array(); //stworz nowa tablice conversions typu JSON
          printf("Nie znaleziono historii operacji\n");
   }

   do //petla MENU GLOWNEGO
   {
          printf("MENU GLOWNE\n\nWcisnij a dla konwersji DEC -> OCT\nWcisnij b dla konwersji OCT -> DEC\nWcisnij c dla konwersji BIN -> HEX\nWcisnij d dla konwersji HEX -> BIN\nWcisnij h, aby wyswietlic historie operacji\nWcisnij s, aby przelaczyc autozapis do pliku\nWcisnij w, aby zapisac i wyjsc z programu\n");
          if (autoSave) printf("\nAutozapis ON\n");
          else printf("\nAutozapis OFF\n");
          printf("Historia %d/20\n", count);

          //pobranie znaku od uzytkownika i wykonanie instrukcji
          userInput = _getch();
          switch (userInput)
          {
```

```
		case 'a': {if (count >= 20) { system("cls"); printf("Osiagnieto limit pamieci, konieczne jest usuniecie dowolnej pozycji z historii\n\n"); } else dec_to_oct(); break; } //DEC->OCT
		case 'b': {if (count >= 20) { system("cls"); printf("Osiagnieto limit pamieci, konieczne jest usuniecie dowolnej pozycji z historii\n\n"); } else oct_to_dec(); break; } //OCT->DEC
		case 'c': {if (count >= 20) { system("cls"); printf("Osiagnieto limit pamieci, konieczne jest usuniecie dowolnej pozycji z historii\n\n"); } else bin_to_hex(); break; } //BIN->HEX
		case 'd': {if (count >= 20) { system("cls"); printf("Osiagnieto limit pamieci, konieczne jest usuniecie dowolnej pozycji z historii\n\n"); } else hex_to_bin(); break; } //HEX->BIN
		case 'h': {view(); break;} //wyswietl historie operacji
		case 's': {if (autoSave) { system("cls"); autoSave = false; printf("Wylaczam autozapis...\n\n"); } else { system("cls"); autoSave = true; printf("Wlaczam autozapis...\n\n"); } break;} //toggle autozapisu
		case 'w': {printf("\nKoncze dzialanie programu...\n"); break;} //zapisz i wylacz program
		default: {system("cls"); printf("Podano niewlasciwy znak\n\n"); break;} //niewlasciwy znak
		}
	} while (userInput != 'w');

	save();
	return 0;
}


//FUNCTION DEFINITIONS
void view() //wyswietla historie operacji i pozwala na usuwanie z niej elementow
{
	system("cls");
	if (count == 0) //jezeli brak rekordow
	{
		printf("Brak rekordow do wyswietlenia\nWracam do menu...\n\n");
		return;
	}

	char userInput; //@param {char} userInput - pobrany od uzytkownika znak
	int i, j = 0; //@param {int} i, j - liczniki do roznych iteracji

	do //petla HISTORII OPERACJI
	{
		//zaznacz wybrana
		conversion = json_object_array_get_idx(conversions, j); //konwersja jako element konwersji o indeksie j
		json_object_object_get_ex(conversion, "input", &input); //pobierz input z konwersji i przypisz do input
		json_object_object_get_ex(conversion, "output", &output); //pobierz output z konwersji i przypisz do output
		json_object_object_get_ex(conversion, "type", &type); //pobierz type z konwersji i przypisz do type

		printf("HISTORIA OPERACJI\n\nObecnie wybrana operacja jest to operacja o numerze #%d\nKonwersja %s liczby %s z wynikiem %s\n\nWcisnij u, aby usunac rekord tej operacji\nWcisnij n, aby wybrac nastepna\nWcisnij p, aby wybrac poprzednia\nWcisnij m, aby wrocic do menu\n\nCala historia %d operacji wyglada nastepujaco:\n", j + 1, json_object_get_string(type), json_object_get_string(input), json_object_get_string(output), count);

		//wyswietlenie wszystkich operacji z ich numerami
		for (i = 0; i < count; i++)
		{
			conversion = json_object_array_get_idx(conversions, i);
			json_object_object_get_ex(conversion, "input", &input);
			json_object_object_get_ex(conversion, "output", &output);
			json_object_object_get_ex(conversion, "type", &type);
```

```
                if (i + 1 < 10) printf("#0%d ", i + 1);
                else printf("#%d ", i + 1);
                printf("Konwersja %s liczby %s z wynikiem %s\n", json_object_get_string(type),
json_object_get_string(input), json_object_get_string(output));
        }

        //pobranie znaku od uzytkownika i wykonanie instrukcji
        userInput = _getch();
        switch (userInput)
        {
                case 'n': //wybierz nastepna
                {
                        system("cls");
                        if (j < count - 1)
                        {
                                printf("Wyznaczanie nastepnej operacji...\n\n");
                                j++;
                        }
                        else printf("Nie ma rekordow nowszych operacji\n\n");
                        break;
                }
                case 'p': //wybierz poprzednia
                {
                        system("cls");
                        if (j > 0)
                        {
                                j--;
                                printf("Wyznaczanie poprzedniej operacji...\n\n");
                        }
                        else printf("Nie ma rekordow starszych operacji\n\n");
                        break;
                }
                case 'u': //usun wybrany rekord
                {
                        system("cls");
                        json_object_array_del_idx(conversions, j, 1);
                        count--;

                        if (count == 0) //jezeli usunieto ostatni rekord, wroc do menu
                        {
                                printf("Usuwam operacje...\nBrak rekordow do wyswietlenia\nWracam
do menu...\n\n");
                                if (autoSave) save();
                                return;
                        }
                        else
                        {
                                printf("Usuwam operacje...\n\n");
                                if (autoSave) save();
                                j = 0;
                                break;
                        }
                }
                case 'm': //wroc do menu
                {
                        break;
                }
                default: //niewlasciwy znak
                {
                        system("cls");
                        printf("Podano niewlasciwy znak\n\n");
                        break;
                }
        }
```

```
   } while (userInput != 'm');

   system("cls");
   printf("Wracam do menu...\n\n");
    return;
}

void insert(char newInput[], char newOutput[], char newType[]) //insert dodaje do tablicy typu
JSON nowy element o parametrach @param {string} newInput, newOutput, newType - kolejno tekst
wejsciowy operacji, wyjsciowy i jej typ
{
   //skracam newInput do ostatniego znaku
   int i = 0; //@param {int} i - licznik do roznych iteracji
   while (newInput[i] != '\n')
   {
          i++;
          if (i >= 49) break; //fallback, gdyby z jakiegos powodu nie odnalazlo znaku newline
   }
   newInput[i] = NULL;

   //skracam newOutput do ostatniego znaku
   i = 0;
   while (newOutput[i] != '\n')
   {
          i++;
          if (i >= 49) break; //fallback, gdyby z jakiegos powodu nie odnalazlo znaku newline
   }
   newOutput[i] = NULL;


   conversion = json_object_new_object(); //tworze nowy objekt konwersji
   json_object_array_add(conversions, conversion); //dodaje go do tablicy

   //przepisuje ze stringow jako nowe obiekty typu JSON (AKA Zbigniew)
   input = json_object_new_string(newInput);
   output = json_object_new_string(newOutput);
   type = json_object_new_string(newType);

   //przypisuje do obiektu "conversion" swoje nowe dane
   json_object_object_add(conversion, "input", input);
   json_object_object_add(conversion, "output", output);
   json_object_object_add(conversion, "type", type);


   count++; //zwieksz liczebnosc
   if (autoSave) save(); //zapisz, jezeli autoSave == true
   return;
}

void save() //zapisuje stan historii operacji w formacie JSON do pliku operationHistory.json,
nadpisujac caly obecny tekst
{
   int i; //@param {int} i - licznik do roznych iteracji
   if (fopen_s(&fp, "operationHistory.json", "w") == 0) //jeżeli pomyślnie wczytano historię
operacji, wypisz komunikat i przepisz historie w formacie JSON
   {
          printf("Zapisywanie historii operacji...\n\n");
          fprintf(fp,"{\n\t\"conversions\":\n\t["); //poczatkowy tekst

          for (i = 0; i < count; i++) //petla do poszczegolnych konwersji
          {
                 conversion = json_object_array_get_idx(conversions, i); //konwersja jako element
konwersji o indeksie i
                 json_object_object_get_ex(conversion, "input", &input); //pobierz input z
konwersji i przypisz do input
```

```
                json_object_object_get_ex(conversion, "output", &output); //pobierz output z
konwersji i przypisz do output
                json_object_object_get_ex(conversion, "type", &type); //pobierz type z konwersji
i przypisz do type

                //szyfrowanie kazdej z wartosci i wpisywanie do pliku
                cipher(json_object_get_string(input), false);
                fprintf(fp, "\n\t\t{\n\t\t\t\"input\": \"%s\",", cipher_out);

                cipher(json_object_get_string(output), false);
                fprintf(fp, "\n\t\t\t\"output\": \"%s\",", cipher_out);

                cipher(json_object_get_string(type), false);
                fprintf(fp, "\n\t\t\t\"type\": \"%s\"\n\t\t}", cipher_out);

                if (i < count - 1) fprintf(fp, ","); //wstaw przecinek, jezeli nie jest ostatnia
        }

        fprintf(fp, "\n\t]\n}"); //koncowy tekst
   }
   else printf("Nie udalo sie otworzyc pliku i zapisac\n\n");

   fclose(fp);
   return;
}

void cipher(char cipher_in[], bool modeDecipher) //funkcja cipher dekoduje lub koduje (zaleznie
od @param {bool} modeDecipher) tekst @param {string} coded, wynikiem jest aktualizacja globalnej
tablicy znakow @param {string} cipher_out
{
   //SZYFROWANIE AUTORSKIE
   //kodowanie/dekodowanie ma 3 mozliwosci obliczenia wartosci w zaleznosci od klucza i
parzystosci napotkanego znaku - ten sam znak na roznych pozycjach moze miec rozna wartosc
   //kodowaniu/dekodowaniu podlegaja tylko wartosci input, output, type, reszta tekstu nie
zmienia sie
   //rozmiar kodowanego tekstu nie zmienia sie

   int key = 0; //@param {int} key - licznik do iteracji, uzyty przy dekodowaniu tekstu
   while (cipher_in[key] != NULL && key < 49) //nie chcemy zamienic NULLa, a jezeli by go nie
bylo z jakiegos dziwnego powodu, to chcemy uniknac nieskonczonej petli
   {
        if ((key % 12 == 0 || key % 12 == 4 || key % 12 == 8) && modeDecipher)
        {
                if (cipher_in[key] % 2 == 0) cipher_out[key] = cipher_in[key] + 5; //dekodowanie
typ #1
                else cipher_out[key] = cipher_in[key] - 7; //dekodowanie typ #2
        }

        else if ((key % 12 == 0 || key % 12 == 4 || key % 12 == 8) && !modeDecipher)
        {
                if (cipher_in[key] % 2 == 1) cipher_out[key] = cipher_in[key] - 5; //kodowanie
typ #1
                else cipher_out[key] = cipher_in[key] + 7; //kodowanie typ #2
        }

        else if(modeDecipher) cipher_out[key] = cipher_in[key] - key % 12; //dekodowanie typ #3
        else cipher_out[key] = cipher_in[key] + key % 12; //kodowanie typ #3
        key++;
   }
   cipher_out[key] = NULL; //oznacz koniec stringa

   return;
}

void dec_to_oct() //konwersja z decymalnego na oktalny
{
```

```
   unsigned long dec; //@param {unsigned long} dec - pobrana od uzytkownika liczba decymalna
   int i, j; //@param {int} i, j - liczniki do roznych iteracji
   char* octReversed, * oct, * decChar; //@param {char*} octReversed, oct, decChar - kolejno
wskazniki na tablice znakow dla liczb oktalnej odwroconej, oktalnej i decymalnej
   int* bin; //@param {int*} bin - wskaznik na tablice liczb dla liczby binarnej

   //inicjalizuje zerami... nieznacznie wieksze dlugosci, niz potrzebne, dla bezpieczenstwa
   bin = (int*)calloc(24, sizeof(int));
   decChar = (char*)calloc(7, sizeof(char));
   oct = (char*)calloc(8, sizeof(char));
   octReversed = (char*)calloc(8, sizeof(char));

   system("cls");

   if (decChar == NULL || oct == NULL || octReversed == NULL || bin == NULL)
   {
          printf("Nie udalo sie alokowac pamieci dla programu\n\n");
          free(decChar);
          free(oct);
          free(octReversed);
          free(bin);
          return;
   }

   bool cyfraZnaczaca = false; //@param {bool} cyfraZnaczaca - mowi o tym, czy dotarto do
pierwszej cyfry znaczacej

   //pobieranie liczby od uzytkownika
   printf("Podaj liczbe dziesietna calkowita nieujemna do konwersji (do 5 znakow)\n");
   scanf_s("%lu", &dec);
   while (getchar() != '\n'); //wylapuje znak newline po uzyciu scanf_s

   if (dec > 99999) //za duza liczba - wiecej niz 5 znakow (tyle wystarczy do objecia wczesniej
ustalonej dla tego podprogramu)
   {
          printf("\nPodano za duzo znakow\n\n");

          free(decChar);
          free(oct);
          free(octReversed);
          free(bin);
          return;
   }

   sprintf_s(decChar, 7, "%lu\n", dec);

   //W tym fragmencie programu zajme sie konwersja z dziesietnego na binarny, z ktorego
przekonwertuje na oktalny
   for (i = 24; i >= 0; i--)
   {
          if (dec >= pow(2, i))
          {
                 dec = dec - pow(2, i);
                 bin[i] = 1;
          }
   }

   //Przedstawienie wyniku
   printf("\nTwoja liczba w systemie osemkowym to ");
   for (i = 7; i > 0; i--)
   {
          octReversed[i - 1] = (char)(bin[(i - 1) * 3] + bin[(i - 1) * 3 + 1] * 2 + bin[(i - 1) *
3 + 2] * 4) + '0';
          if (octReversed[i - 1] != '0') cyfraZnaczaca = true;
          if (cyfraZnaczaca)
          {
```

```
                printf("%c", octReversed[i - 1]);
        }
    }
    i = 0;
    j = 7;
    cyfraZnaczaca = false;

    while (j >= 0) //przepisanie octReversed na oct
    {
        if (!cyfraZnaczaca && octReversed[j] >= '1' && octReversed[j] <= '7') cyfraZnaczaca =
true;
        if (cyfraZnaczaca)
        {
            oct[i] = octReversed[j];
            i++;
        }
        j--;
    }
    printf("\n\n");

    free(bin);
    free(octReversed);
    insert(decChar, oct, "DEC -> OCT");
    free(decChar);
    free(oct);
    return;
}

void oct_to_dec() //konwersja z oktalnego na decymalny
{
    unsigned long octInt, dec = 0; //@param {unsigned long} octInt, dec - kolejno liczba osemkowa
podana przez uzytkownika i decymalna po konwersji
    int i = 0; //@param {int} i - licznik do roznych iteracji
    int* octReversed; //@param {int*} octReversed - wskaznik na tablice liczb dla oktalnej
odwroconej
    char* oct, * decChar; //@param {char*} oct, decChar - kolejno wskazniki na tablice znakow dla
oktalnej i decymalnej

    //inicjalizuje zerami... nieznacznie wieksze dlugosci, niz potrzebne, dla bezpieczenstwa
    decChar = (char*)calloc(8, sizeof(char));
    oct = (char*)calloc(8, sizeof(char));
    octReversed = (int*)calloc(8, sizeof(int));

    system("cls");

    if (decChar == NULL || oct == NULL || octReversed == NULL)
    {
        printf("Nie udalo sie alokowac pamieci dla programu\n\n");
        free(decChar);
        free(oct);
        free(octReversed);
        return;
    }

    //pobieranie liczby od uzytkownika
    printf("Podaj liczbe osemkowa calkowita nieujemna do konwersji (do 6 znakow)\n");
    scanf_s("%lu", &octInt);

    if (octInt > 777777) //za duza liczba - wiecej niz 6 znakow (tyle wystarczy do objecia
wczesniej ustalonej dla tego podprogramu)
    {
        printf("\nPodano za duzo znakow\n\n");

        free(decChar);
        free(oct);
        free(octReversed);
```

```
		return;
	}

	while (getchar() != '\n');
	sprintf_s(oct, 8, "%lu\n", octInt); //przepisuje octInt (int) na oct (char)


	//W tym fragmencie programu przepisze podana liczbe oktalna octInt do tablicy octReversed
	while (octInt > 0)
	{
		octReversed[i] = octInt % 10;
		if (octReversed[i] >= 8)
		{
			printf("\nPodano niewlasciwy znak\n\n");
			free(decChar);
			free(oct);
			free(octReversed);
			return;
		}
		octInt = octInt / 10;
		i++;
	}

	//Zamieniam liczbe oktalna na decymalna
	for (i = 0; i < 7; i++)
	{
		dec += octReversed[i] * pow(8, i);
	}

	//Przedstawienie wyniku
	printf("\nTwoja liczba w systemie dziesietnym to %d\n\n", dec);
	sprintf_s(decChar, 8, "%lu", dec); //przepisuje dec (int) na decChar (char)

	free(octReversed);
	insert(oct, decChar, "OCT -> DEC");
	free(oct);
	free(decChar);
	return;
}

void bin_to_hex() //konwersja z binarnego na heksadecymalny
{
	int i = 0, j = 0; //@param {int} i, j - liczniki do roznych iteracji
	char* bin, * hex; //@param {char*} bin, hex - kolejno wskazniki na tablice znakow dla binarnej
i heksadecymalnej
	int* hexReversed; //@param {int*} hexReversed - wskaznik na tablice liczb dla odwroconej
heksadecymalnej

	//inicjalizuje zerami... nieznacznie wieksze dlugosci, niz potrzebne, dla bezpieczenstwa
	bin = (char*)calloc(34, sizeof(char));
	hexReversed = (int*)calloc(9, sizeof(int));
	hex = (char*)calloc(9, sizeof(char));

	system("cls");

	if (bin == NULL || hexReversed == NULL || hex == NULL)
	{
		printf("Nie udalo sie alokowac pamieci dla programu\n\n");
		free(bin);
		free(hexReversed);
		free(hex);
		return;
	}

	//POBRANIE LICZBY DO KONWERSJI
	printf("Podaj liczbe dwojkowa do konwersji (do 31 znakow)\n");
```

```
	for (i = 0; i < 33; i++)
	{
		scanf_s("%c", &bin[i]);

		if (bin[i] == '\n') break; //jesli koniec, wyjdz z petli
		else if (bin[i] != '1' && bin[i] != '0' && bin[i] != '\n') //jesli podano znak inny od
1, 0, \n, zakoncz dzialanie funkcji
		{
			printf("\nPodano niewlasciwy znak\n\n");
			while (getchar() != '\n');
			free(bin);
			free(hexReversed);
			free(hex);
			return;
		}
	}

	if (i == 33) //za duza liczba - wiecej niz 31 znakow (tyle wystarczy do objecia wczesniej
ustalonej dla tego podprogramu)
	{
		if (bin[i] != '\n') while (getchar() != '\n');
		printf("\nPodano za duzo znakow\n\n");
		free(bin);
		free(hexReversed);
		free(hex);
		return;
	}

	//KONWERSJA Z BIN NA HEX
	i--; //ustaw @param {int} i na ostatni znak (przed \n)
	j = 0;
	while (i >= 0)
	{
		if (bin[i] == '1') hexReversed[j] += 1;
		if (bin[i - 1] == '1') hexReversed[j] += 2;
		if (bin[i - 2] == '1') hexReversed[j] += 4;
		if (bin[i - 3] == '1') hexReversed[j] += 8;
		j++;
		i -= 4;
	}

	//WYPISANIE LICZBY HEX
	j--;
	i = 0;
	printf("\nTwoja liczba po konwersji to ");
	while (j >= 0)
	{
		if (hexReversed[j] <= 9) { hex[i] = (char)hexReversed[j] + '0'; printf("%d",
hexReversed[j]); }
		else if (hexReversed[j] == 10) { hex[i] = 'A'; printf("%c", 'A'); }
		else if (hexReversed[j] == 11) { hex[i] = 'B'; printf("%c", 'B'); }
		else if (hexReversed[j] == 12) { hex[i] = 'C'; printf("%c", 'C'); }
		else if (hexReversed[j] == 13) { hex[i] = 'D'; printf("%c", 'D'); }
		else if (hexReversed[j] == 14) { hex[i] = 'E'; printf("%c", 'E'); }
		else if (hexReversed[j] == 15) { hex[i] = 'F'; printf("%c", 'F'); }
		j--;
		i++;
	}
	printf("\n\n");

	free(hexReversed);
	insert(bin, hex, "BIN -> HEX");
	free(bin);
	free(hex);

	return;
```

```
}

void hex_to_bin() //konwersja z heksadecymalnego na binarny
{
   int i = 0, j = 0; //@param {int} i, j - liczniki do roznych iteracji
   char* hex, * bin; //@param {char*} hex, bin - kolejno wskazniki na tablice znakow dla
heksadecymalnej i binarnej

   //inicjalizuje zerami... nieznacznie wieksze dlugosci, niz potrzebne, dla bezpieczenstwa
   hex = (char*)calloc(10, sizeof(char));
   bin = (char*)calloc(34, sizeof(char));

   system("cls");

   if (hex == NULL || bin == NULL)
   {
          printf("Nie udalo sie alokowac pamieci dla programu\n\n");
          free(hex);
          free(bin);
          return;
   }

   //POBRANIE LICZBY DO KONWERSJI
   printf("Podaj liczbe szesnastkowa do konwersji (do 8 znakow)\n");
   for (i = 0; i < 9; i++)
   {
          scanf_s("%c", &hex[i]);

          if (hex[i] == '\n') break; //jesli koniec, wyjdz z petli
          else if (((int)hex[i] < 48 || (int)hex[i] > 57) && ((int)hex[i] < 65 || (int)hex[i] >
70) && (int)hex[i] != '\n') //jesli podano znak inny od 0-9, A, B, C, D, E, F, \n, zakoncz
dzialanie funkcji
          {
                 printf("\nPodano niewlasciwy znak\n\n");
                 while (getchar() != '\n');
                 free(hex);
                 free(bin);
                 return;
          }
   }

   if (i == 9) //za duza liczba - wiecej niz 8 znakow (tyle wystarczy do objecia wczesniej
ustalonej dla tego podprogramu)
   {
          if (hex[i] != '\n') while (getchar() != '\n');
          printf("\nPodano za duzo znakow\n\n");
          free(bin);
          free(hex);
          return;
   }

   //KONWERSJA Z HEX NA BIN (WYPISANIE)
   printf("\nTwoja liczba po konwersji to \n");
   j = i - 1;
   i = 0;
   while (i <= j)
   {
          if (hex[i] == '0') { printf("0000"); strcat_s(bin, 34, "0000"); }
          else if (hex[i] == '1') { printf("0001"); strcat_s(bin, 34, "0001"); }
          else if (hex[i] == '2') { printf("0010"); strcat_s(bin, 34, "0010"); }
          else if (hex[i] == '3') { printf("0011"); strcat_s(bin, 34, "0011"); }
          else if (hex[i] == '4') { printf("0100"); strcat_s(bin, 34, "0100"); }
          else if (hex[i] == '5') { printf("0101"); strcat_s(bin, 34, "0101"); }
          else if (hex[i] == '6') { printf("0110"); strcat_s(bin, 34, "0110"); }
          else if (hex[i] == '7') { printf("0111"); strcat_s(bin, 34, "0111"); }
          else if (hex[i] == '8') { printf("1000"); strcat_s(bin, 34, "1000"); }
```

```
            else if (hex[i] == '9') { printf("1001"); strcat_s(bin, 34, "1001"); }
            else if (hex[i] == 'A') { printf("1010"); strcat_s(bin, 34, "1010"); }
            else if (hex[i] == 'B') { printf("1011"); strcat_s(bin, 34, "1011"); }
            else if (hex[i] == 'C') { printf("1100"); strcat_s(bin, 34, "1100"); }
            else if (hex[i] == 'D') { printf("1101"); strcat_s(bin, 34, "1101"); }
            else if (hex[i] == 'E') { printf("1110"); strcat_s(bin, 34, "1110"); }
            else if (hex[i] == 'F') { printf("1111"); strcat_s(bin, 34, "1111"); }
            i++;
    }
    printf("\n\n");

    insert(hex, bin, "HEX -> BIN");
    free(hex);
    free(bin);
    return;
}
```

Screeny z działania programu

Pierwsze otwarcie programu

```
Nie znaleziono historii operacji
MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 0/20
```

<wejście>

```
Podaj liczbe dziesietna calkowita nieujemna do konwersji (do 5 znakow)
123456

Podano za duzo znakow

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 0/20
```

a

<liczba>

<enter>

```
Podaj liczbe dziesietna calkowita nieujemna do konwersji (do 5 znakow)
12345

Twoja liczba w systemie osemkowym to 30071

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 1/20
```

a

<liczba>

<enter>

```
Podaj liczbe osemkowa calkowita nieujemna do konwersji (do 6 znakow)
1234567

Podano za duzo znakow

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 1/20
```

b

<liczba>

<enter>

```
Podaj liczbe osemkowa calkowita nieujemna do konwersji (do 6 znakow)
2468

Podano niewlasciwy znak

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 1/20
```

b

<liczba>

<enter>

```
Podaj liczbe osemkowa calkowita nieujemna do konwersji (do 6 znakow)
123456

Twoja liczba w systemie dziesietnym to 42798

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 2/20
```

b

<liczba>

<enter>

```
Podaj liczbe dwojkowa do konwersji (do 31 znakow)
10101010101010101010101010100101010

Podano za duzo znakow

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 2/20
```

c
<liczba>
<enter>

```
Podaj liczbe dwojkowa do konwersji (do 31 znakow)
10101010101010101010010121010

Podano niewlasciwy znak

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 2/20
```

c
<liczba>
<enter>

```
Podaj liczbe dwojkowa do konwersji (do 31 znakow)
10101010111101010101111010101100

Twoja liczba po konwersji to AAF55EAC

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 3/20
```

c
<liczba>
<enter>

```
Podaj liczbe szesnastkowa do konwersji (do 8 znakow)
123456789

Podano za duzo znakow

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 3/20
```

d

<liczba>

<enter>

```
Podaj liczbe szesnastkowa do konwersji (do 8 znakow)
ABCDEF98

Twoja liczba po konwersji to
10101011110011011110111110011000

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 4/20
```

d

<liczba>

<enter>

```
Podaj liczbe dziesietna calkowita nieujemna do konwersji (do 5 znakow)
75082

Twoja liczba w systemie osemkowym to 222512

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 5/20
```

a

<liczba>

<enter>

```
Wlaczam autozapis...

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis ON
Historia 5/20
```

s

```
Podaj liczbe osemkowa calkowita nieujemna do konwersji (do 6 znakow)
222512

Twoja liczba w systemie dziesietnym to 75082

Zapisywanie historii operacji...

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis ON
Historia 6/20
```

b

<liczba>

<enter>

```
Podaj liczbe dwojkowa do konwersji (do 31 znakow)
11111111111111101101101010110101

Twoja liczba po konwersji to 7FFEDAB5

Zapisywanie historii operacji...

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis ON
Historia 7/20
```

c

<liczba>

<enter>

```
Podaj liczbe szesnastkowa do konwersji (do 8 znakow)
7FFEDAB5

Twoja liczba po konwersji to
01111111111111101101101010110101

Zapisywanie historii operacji...

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis ON
Historia 8/20
```

d

<liczba>

<enter>

```
Wylaczam autozapis...

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 8/20

Koncze dzialanie programu...
Zapisywanie historii operacji...


C:\Users\coman\Desktop\WDP (PD na 16 czerwca)\wdpjson\Debug\wdpjson.exe (proces 12700) z
akończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

s

w

operationHistory.json po pierwszym zamknięciu programu

Drugie otwarcie programu

```
Pomyslnie wczytano historie operacji
Zawiera ona nastepujaca liczbe zapisow: 8

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 8/20
```

```
HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #1
Konwersja DEC -> OCT liczby 12345 z wynikiem 30071

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu
```

h

```
Cala historia 8 operacji wyglada nastepujaco:
#01 Konwersja DEC -> OCT liczby 12345 z wynikiem 30071
#02 Konwersja OCT -> DEC liczby 123456 z wynikiem 42798
#03 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#04 Konwersja HEX -> BIN liczby ABCDEF98 z wynikiem 10101011110011011110111110011000
#05 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#06 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#07 Konwersja BIN -> HEX liczby 11111111111111011011010101101 z wynikiem 7FFEDAB5
#08 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111110110110101011010 1
```

```
Usuwam operacje...

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #1
Konwersja OCT -> DEC liczby 123456 z wynikiem 42798

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu
```

u

```
Cala historia 7 operacji wyglada nastepujaco:
#01 Konwersja OCT -> DEC liczby 123456 z wynikiem 42798
#02 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#03 Konwersja HEX -> BIN liczby ABCDEF98 z wynikiem 10101011110011011110111110011000
#04 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#05 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#06 Konwersja BIN -> HEX liczby 11111111111111011011010101101 z wynikiem 7FFEDAB5
#07 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111110110110101011010 1
```

```
Wyznaczanie nastepnej operacji...

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #2
Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu

Cala historia 7 operacji wyglada nastepujaco:
#01 Konwersja OCT -> DEC liczby 123456 z wynikiem 42798
#02 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#03 Konwersja HEX -> BIN liczby ABCDEF98 z wynikiem 10101011110011011110111110011000
#04 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#05 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#06 Konwersja BIN -> HEX liczby 11111111111111101101101010110101 z wynikiem 7FFEDAB5
#07 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111101101101010110101
```

n

```
Wyznaczanie nastepnej operacji...

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #3
Konwersja HEX -> BIN liczby ABCDEF98 z wynikiem 10101011110011011110111110011000

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu

Cala historia 7 operacji wyglada nastepujaco:
#01 Konwersja OCT -> DEC liczby 123456 z wynikiem 42798
#02 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#03 Konwersja HEX -> BIN liczby ABCDEF98 z wynikiem 10101011110011011110111110011000
#04 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#05 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#06 Konwersja BIN -> HEX liczby 11111111111111101101101010110101 z wynikiem 7FFEDAB5
#07 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111101101101010110101
```

n

```
Usuwam operacje...

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #1
Konwersja OCT -> DEC liczby 123456 z wynikiem 42798

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu

Cala historia 6 operacji wyglada nastepujaco:
#01 Konwersja OCT -> DEC liczby 123456 z wynikiem 42798
#02 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#03 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#04 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#05 Konwersja BIN -> HEX liczby 11111111111111101101101010110101 z wynikiem 7FFEDAB5
#06 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111101101101010110101
```

u

```
Nie ma rekordow starszych operacji

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #1
Konwersja OCT -> DEC liczby 123456 z wynikiem 42798

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu

Cala historia 6 operacji wyglada nastepujaco:
#01 Konwersja OCT -> DEC liczby 123456 z wynikiem 42798
#02 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#03 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#04 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#05 Konwersja BIN -> HEX liczby 11111111111111101101101010110101 z wynikiem 7FFEDAB5
#06 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111101101101010110101
```

p

```
Usuwam operacje...

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #1
Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu

Cala historia 5 operacji wyglada nastepujaco:
#01 Konwersja BIN -> HEX liczby 10101010111101010101111010101100 z wynikiem AAF55EAC
#02 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#03 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#04 Konwersja BIN -> HEX liczby 11111111111111101101101010110101 z wynikiem 7FFEDAB5
#05 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111101101101010110101
```

u

```
Usuwam operacje...

HISTORIA OPERACJI

Obecnie wybrana operacja jest to operacja o numerze #1
Konwersja DEC -> OCT liczby 75082 z wynikiem 222512

Wcisnij u, aby usunac rekord tej operacji
Wcisnij n, aby wybrac nastepna
Wcisnij p, aby wybrac poprzednia
Wcisnij m, aby wrocic do menu

Cala historia 4 operacji wyglada nastepujaco:
#01 Konwersja DEC -> OCT liczby 75082 z wynikiem 222512
#02 Konwersja OCT -> DEC liczby 222512 z wynikiem 75082
#03 Konwersja BIN -> HEX liczby 11111111111111101101101010110101 z wynikiem 7FFEDAB5
#04 Konwersja HEX -> BIN liczby 7FFEDAB5 z wynikiem 01111111111111101101101010110101
```

u

```
Wracam do menu...

MENU GLOWNE

Wcisnij a dla konwersji DEC -> OCT
Wcisnij b dla konwersji OCT -> DEC
Wcisnij c dla konwersji BIN -> HEX
Wcisnij d dla konwersji HEX -> BIN
Wcisnij h, aby wyswietlic historie operacji
Wcisnij s, aby przelaczyc autozapis do pliku
Wcisnij w, aby zapisac i wyjsc z programu

Autozapis OFF
Historia 4/20

Koncze dzialanie programu...
Zapisywanie historii operacji...

C:\Users\coman\Desktop\WDP (PD na 16 czerwca)\wdpjson\Debug\wdpjson.exe (proces 13760) z
akończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

m

w

operationHistory.json po drugim zamknięciu programu

operationHistory.json | operationHistory — kopia.json | wdpjson.c

Schemat: <Nie wybrano schematu>


```
{
    "conversions":
    [
        {
            "input": "262;9",
            "output": "9348,7",
            "type": "KFE#(C&V>]"
        },
        {
            "input": "9348,7",
            "output": "262;9",
            "type": "JDV#(C&K@L"
        },
        {
            "input": ",234,678,:;<,224,5787::<7233,57",
            "output": "2GHHKFH<",
            "type": "IJP#(C&O@a"
        },
        {
            "input": "2GHHKFH<",
            "output": "7234,678,:;<,233,668,9;;,1347668",
            "type": "OFZ#(C&IDW"
        }
    ]
}
```