Wojskowa Akademia Techniczna
Wydział Elektroniki
Instytut Telekomunikacji

Studia I°

Języki C/C++ w zastosowaniach sieciowych

Materiały pomocnicze do zajęć
Dodatek 1

dr inż. Jarosław Krygier

Warszawa 2020

Spis treści

Zawartość

Program W0

```
/*
 ============================================================================
 Name        : W0.c
 Author      : Jaroslaw Krygier, Wojskowa Akademia Techniczna, WEL
 Version     : v1
 Copyright   : Kopiowanie i wykorzystywanie bez zgody wlasciciela zabronione. Mozna uzywac jako referencja
jedynie w ramach zajec.
 Description : Przedmiot: Języki C/C++ W zastosowaniach sieciowych - W0
 ============================================================================
 */

/* Instrukcje preprocesora*/
#include <stdio.h>
#include <stdlib.h>

// Deklaracja zmiennej globalnej
int zmienna_globalna = 10;

/*Deklaracja funkcji*/
int wypisz ();
int wypisz_2 ();

/*Program glowny*/
int main(void) {
    puts("W0-Struktura aplkacji w C"); /* prints W0 */
    // Deklaracja zmiennych lokalnych programu glownego
    int a =1, b=2;
    int c;
    //cialo programu glownego
    //listowanie zmiennej globalnej
    printf ("Zmienna globalna->funkcja glowna: %d \n", zmienna_globalna);

    //Funkcja 1
    c = wypisz(a, b);
    printf ("Wynik fukcji_1: %d \n", c);
    //Fukcja 2
    if (wypisz_2()==1) {
        //funkcja wykonana prawidlowo
        printf ("Funkcja_2 wykonana prawidlowo");
    }
    else {
        //funkcja wykonana nieprawidlowo: blad!!!
        printf ("Funkcja_2 zwrociala blad");
    }

    return EXIT_SUCCESS;
}


/*Funkcje*/

int wypisz (int i, int j) {
    //deklaracja zmiennych lokalnych funkcji
    int k;

    //cialo funkcji
    printf ("Wykonuje funkcje_1 'wypisz'\n");
    printf ("Zmienna globalna->funkcja_1: %d \n", zmienna_globalna);
    k = i+j;

    //zmienna zwracana
    return (k);
}

int wypisz_2 (void) {
    // Czesto wartos zwracana oznacza pozytywny lub negatywny wynik funkcji
    // a uzyteczne dane zwracane sa poprzez wskazniki
    int i=0;
```

```
    printf ("Wykonuje funkcje_2 'wypisz_2'\n");
    printf ("Zmienna globalna->funkcja_2: %d \n", zmienna_globalna);

    if (i==0) {
        return (-1);
    }
    else {
        return (1);
    }
}
```

Program W1

```
/*
 ============================================================================
 Name        : W1.c
 Author      : Jaroslaw Krygier, Wojskowa Akademia Techniczna, WEL
 Version     : v1
 Copyright   : Kopiowanie i wykorzystywanie bez zgody wlasciciela zabronione. Mozna uzywac jako referencja
jedynie w ramach zajec.
 Description : Przedmiot: Języki C/C++ W zastosowaniach sieciowych - W1
 ============================================================================
 */


#include <stdio.h>
#include <stdlib.h>
#include <endian.h>


int main(){
    puts("W1: Wprowadzenie przypomnienie zasad programowania w jezyku C");

    /*************************************
     * Deklaracja zmiennych - typy zmiennych
     *************************************/
    int             a; //calkowita
    float           b; //zmiennoprzecinkowa pojedynczej precyzji
    double          c; //zmiennoprzecinkowa podwojnej precyzji
    long double     cc; //zmiennoprzecinkowa podwojnej precyzji rozszerzona
    char            d; //znakowa
    short int       e; //mozna uzywac tylko: short
    long int        f; //mozna uzywac tylko: long, =int
    signed int      g; //=int
    unsigned int    h; //int >0
    signed char     i; //=char
    unsigned char       j; //char >0
    int             *adres_int;
    char            *adres_char;
    double          *adres_double;
    unsigned char   tabela[10];

    int             out;

    /*****************************************************************************
     * Rozmiary pamieci w [B] zarezerwoqwnych dla poszczegolnych typow zmiennych
     *****************************************************************************/
    printf ("==========Rozmiary pamieci=========\n");
    out = sizeof (a);
    printf ("Rozmiar int:%d \n", out);
    out = sizeof (b);
    printf ("Rozmiar float:%d \n", out);
    out = sizeof (c);
    printf ("Rozmiar double:%d \n", out);
    out = sizeof (cc);
    printf ("Rozmiar long double:%d \n", out);
    out = sizeof (d);
    printf ("Rozmiar char:%d \n", out);
    out = sizeof (e);
    printf ("Rozmiar short int:%d \n", out);
    out = sizeof (f);
    printf ("Rozmiar long int:%d \n", out);
    out = sizeof (g);
    printf ("Rozmiar signed int:%d \n", out);
    out = sizeof (h);
    printf ("Rozmiar unsigned int:%d \n", out);
    out = sizeof (i);
    printf ("Rozmiar signed char:%d \n", out);
    out = sizeof (j);
    printf ("Rozmiar unsigned char:%d \n", out);
    out = sizeof (adres_int);
    printf ("Rozmiar adres_int:%d \n", out);
```

```
out = sizeof (adres_char);
printf ("Rozmiar adres_char:%d \n", out);
out = sizeof (adres_double);
printf ("Rozmiar adres_double:%d \n", out);
out =sizeof (tabela); //adres
printf ("Rozmiar zarezerwowanej pamieci dla tabela:%d \n", out);


/*************************************************************
 * Forma zapisu danych
 * ********************
 * Little endian: zapis liczby '0xaabbccdd' w pamieci: dd cc bb aa
 * Big endian: zapis liczby '0xaabbccdd' w pamieci: aa bb cc dd
 *************************************************************/
printf ("==========Forma zapisu danych==========\n");
//deklaracja
unsigned int zmienna;
struct oktety {
    unsigned char oktet1;
    unsigned char oktet2;
    unsigned char oktet3;
    unsigned char oktet4;
};
struct oktety zmienna1;
//koniec deklaracji

zmienna=0xaabbccdd; //zmianna typu int (4B)

memcpy (&zmienna1, &zmienna, 4); // kopia 4 oktetow ze zmienna (int) do zmienna1 (struct)

printf ("Zawartosc zmienna: 0x%x, liczba bajtow w 'zmianna': %d, ulozenie bajtow: [%x, %x, %x, %x] \n",
        zmienna, sizeof (zmienna), zmienna1.oktet1, zmienna1.oktet2, zmienna1.oktet3, zmienna1.oktet4);
//Zatem forma zapisu to:....?

//Dodatkowo sprawdzamy z wykorzystaniem pliku naglowkowego: <endian.h>
if (__BYTE_ORDER == __BIG_ENDIAN)
    printf("Porządek big-endian (1234)\n");
else if (__BYTE_ORDER == __LITTLE_ENDIAN)
    printf("Porządek little-endian (4321)\n");
else
    printf("Inny porządek (%d)\n", __BYTE_ORDER);


/*********************************************
 *Struktury
 *********************************************/
printf ("==========Struktury==========\n");

//>>>>>>>>>>>>>>>>>>przyklad 1
//deklaracja
struct struktura1 {
    unsigned int a;
    unsigned int b;
    unsigned int c;
} zmienna_structura1;
//koniec deklaracji struktury

int rozmiar_pamieci;

//rozmiar zarezerwowanej pamieci
rozmiar_pamieci = sizeof(zmienna_structura1);
printf ("Rozmian struktury1 (int, int, int): %d \n", rozmiar_pamieci);

//>>>>>>>>>>>>>>>>>>przyklad 2
//deklaracja
struct struktura2 {
    unsigned int a;
    unsigned char b;
} zmienna_struktura2;
//koniec deklaracji struktury
```

```
rozmiar_pamieci = sizeof(zmienna_struktura2);
printf ("Rozmian struktury2 (int, char): %d \n", rozmiar_pamieci);

//>>>>>>>>>>>>>>>>>>przyklad 3
//deklaracja
struct struktura3 {
    unsigned char a;
    unsigned char b;
} zmienna_struktura3;
//koniec deklaracji struktury

rozmiar_pamieci = sizeof(zmienna_struktura3);
printf ("Rozmian struktury3 (char, char): %d \n", rozmiar_pamieci);
//>>>>>>>>>>>>>>>>>>przyklad 4
//deklaracja
struct struktura4 {
    unsigned char a;
    short int   b;
    unsigned char c;
} zmienna_struktura4;
//koniec deklaracji struktury

rozmiar_pamieci = sizeof(zmienna_struktura4);
printf ("Rozmian struktury4 (char, short int, char): %d \n", rozmiar_pamieci);


//odwolanie do struktury
zmienna_struktura4.a = 0xff;
zmienna_struktura4.b = 0x1234;
zmienna_struktura4.c = 0xaa;

printf ("Odwolanie do struktury 'struktura4':a: 0x%x, b: 0x%x, c: 0x%x \n", zmienna_struktura4.a,
        zmienna_struktura4.b, zmienna_struktura4.c);

//struktura w strukturze
struct struktura5 {
    unsigned char a;
    unsigned char b;
};

struct struktura6 {
    unsigned int a;
    unsigned int b;
    struct struktura5   c;
};

//deklaracja
struct struktura6   zmienna_struktura6;
//koniec deklaracji

rozmiar_pamieci = sizeof(zmienna_struktura6);
printf ("=========Rozmian struktury6 (int, int, struct (char, char)): %d \n", rozmiar_pamieci);
rozmiar_pamieci = sizeof(zmienna_struktura6.c.a);
printf ("Rozmian struktury6: zmienna_struktura6.c.a (char): %d \n", rozmiar_pamieci);

//odwolanie do stuktury zagniezdzonej
zmienna_struktura6.c.a = 0x12;

//tablica struktur
//deklaracja
struct struktura5 tablica_structur [4];
//koniec deklaracji

rozmiar_pamieci = sizeof(tablica_structur);
printf ("Rozmian tablica_struktur [4]: %d \n", rozmiar_pamieci);
```

```
/************************************************
Opearcje arytmetyczne
+, -, *, /, %
************************************************/

printf ("==========Operacje arytmetyczne==========\n");
//deklaracja
int k, l, m;
//koniec deklaracji

k=11;
l=3;

m=k+l;
printf ("wynik k + l:%d \n", m);
m=k-l;
printf ("wynik k - l:%d \n", m);
m=k*l;
printf ("wynik k * l:%d \n", m);
m=k/l;
printf ("wynik k / l:%d \n", m);
m=k%l;
printf ("wynik k mod l:%d \n", m);

/************************************************
* Relacje i operacje logiczne
************************************************/
/* relacje:              >, >=, <, <=, ==, !=         */
/* operacje logiczne:    &&, ||                       */

printf ("==========Relacje==========\n");
//deklaracje
int     log_a=1;
int     log_b=2;
double  log_c=3.0;
double  log_d=3.0;
//koniec deklaracji

//relacje
if (log_c==log_d){
    printf ("log_c==log_d \n");
}
else {
    printf ("log_c!=log_d \n");
}

if (log_a>=log_b){
    printf ("log_a>=log_b \n");
}
else {
    printf ("log_a<log_b \n");
}

//operacje logiczne
if ((log_a>log_b)||(log_c>log_d)){
    printf ("(log_a>log_b)||(log_c>log_d) \n");
}
else if ((log_a<log_b)&& (log_c==log_d)){
    printf ("(log_a<log_b)&& (log_c==log_d) \n");
}

//kolejnosc czytania: od lewej do prawej
log_a=0;
//przyklad ponizszy nie jest powszechnie stosowany i ze wzgledu na nieczytelnosc uwazany za zly styl
 programowania
if ((log_a>0)||(log_b<0)||(log_a=1)){
    printf("Wynik kierunku czytania od lewej: log_a = %d\n", log_a);
}
else {
    printf("Wynik kierunku czytania od prawej: log_a = %d\n", log_a);
}
```

```
/************************************************
Operator wyrazenia warunkowego
a?b:c
************************************************/
printf ("==========Operator wyrazenia warunkowego==========\n");
log_a=1; log_b=2;
(log_a>log_b) ? printf ("PRAWDA\n") : printf ("FALSZ\n");

//znaczenie: jesli a==PRAWDA => b, jesli a==FALSZ => c

/************************************************************
 * Inkrementacja i dekremantacja
 ************************************************************/
printf ("==========Inkrementacja, dekrementacja==========\n");
//Deklaracja
int o, p;
//Koniec deklaracji

o=10;
p=o++; //postinkrementacja
printf ("postinkrementacja: p = %d, o= %d \n", p, o);
o=10;
p=o--; //postdekremantacja
printf ("postdekremantacja: p = %d, o= %d \n", p, o);
o=10;
p=++o; //preinkrementacja
printf ("preinkrementacja: p = %d, o= %d \n", p, o);
o=10;
p=--o; //predekremantacja
printf ("preinkrementacja: p = %d, o= %d \n", p, o);

/************************************************
 * Operacje bitowe - działania na liczbach calkowitych
 ************************************************/
// negacja bitowa ("~"),
// koniunkcja bitowa ("&"),
// alternatywa bitowa ("|") i
// alternatywa rozłączna (XOR) ("^")

printf ("==========Operacje bitowe==========\n");

//deklaracja
unsigned char    aaa, ddd;
unsigned int bbb, eee;
//koniec deklaracji
//Uwaga: obesrwowac zawartosc zmiennych bitowych w debugerze

//negacja ~
aaa = 0xff;
aaa = ~aaa;          //char
aaa = 0b00001111;
aaa = ~aaa;          //char
bbb = 0xf0f0f0f0;
bbb = ~bbb;          //uint

//koniunkcja (and) &
aaa = 0xff;
ddd = 0xf0;
aaa = aaa & ddd;     //liczby char

aaa = 0xff;
bbb = 0xf0f0f0ff;
eee = bbb & aaa; //char i uint

//przyklad praktyczny: adres ip i maska
bbb = 0x123456ee;    //adres
eee = 0xffff0000;    //maska
bbb = bbb & eee; //adres sieci
```

```
//alternatywa (or) |
aaa = 0xab;
ddd = 0xf0;
aaa = aaa | ddd;  // liczby char

aaa = 0xfe;
bbb = 0xf0f0f0ff;
eee = bbb | aaa; //char i uint

//XOR
aaa = 0xab;
ddd = 0xf0;
eee = aaa ^ ddd;  // liczby char (na eee rzutowana char)

aaa = 0xfe;
bbb = 0xf0f0f0ff;
eee = bbb ^ aaa; //char i uint


/*************************************************
 * Operacje bitowe: przesuniecia
 *************************************************/
printf ("==========Operacje bitowe: przesuniecia==========\n");

//deklaracja
unsigned int ggg, hhh, iii;
unsigned int jjj;
//koniec deklaracji

//w lewo
ggg = 0xff00ff00;
hhh = ggg<<4;
iii = ggg<<8;
jjj = ggg*16;    // (ggg * 2 do potegi 4)
// pszesuniecie w lewo w liczbie x o y bitow rownoznaczne jest pomnozeniu tej liczby przez 2 do potegi x

//w prawo
ggg = 0xff00ff00;
hhh = ggg>>4;
iii = ggg>>8;
jjj = ggg/16; // (ggg / 2 do potegi 4)
//pszesuniecie w prawo w liczbie x o y bitow rownoznaczne jest podzieleniu tej liczby przez 2 do potegi
  x

/*************************************************
* Tabelice
* typ nazwa_tablicy[rozmiar];
* ************************************************/

printf ("==========Tablice==========\n");
//deklaracja
int tablica_1[5]={1,2,3,4,5};
int tablica_2[5]={1,};
int tablica_3[]={1,2,3,4,5,6};
unsigned char tablica_4[4]; //tablice znakowe zwane sa tez buforami
//wielowymiarowa = zwrocic uwage na adresowanie poszczegolnych wierszy
unsigned char tablica_macierz [2][4] = {  {0x01,0x02,0x03,0x04},
                                          {0x05,0x06,0x07,0x08}    };
//koniec deklaracji

//uwaga na zapis zmiennych wielobajtowych do buforow tablicowych - Little Endian
int zmienna_tab =0xaabbccdd;
memcpy (tablica_4, &zmienna_tab, 4);

/*============================================================*/
/* Uwaga!: kompilator gcc w OS Linux umozliwia
 * dokladne przydzielenie liczby bajtow dla struktur zawierajacych
 * zmienne roznych typow, ale trzeba wykorzystac instrukcje:
 * __attribute__ ((packed))
 * lub
 * #pragma pack(1)  ....  #pragma pack()
```

```
     * Nie wszystkie kompilatory jednak mogą tak robic.
     */

    //bez upakowania
    struct struktura7_a {
            unsigned int     a; //4B
            unsigned char    b; //1B
            int c; //4B
        };

    printf ("Wielkosc pamieci zarezerwowanej dla struktury7_a: %d \n", sizeof (struct struktura7_a));


    struct struktura7_b {
        unsigned int     a; //4B
        unsigned char    b; //1B
        int c; //4B
    } __attribute__ ((packed));

    printf ("Wielkosc pamieci zarezerwowanej dla struktury7_b: %d \n", sizeof (struct struktura7_b));

    #pragma pack(1) //wlaczenie upakowania
    struct struktura8 {
            unsigned int     a; //4B
            unsigned char    b; //1B
            int c; //4B
        };
    #pragma pack() //wylaczenie upakowania

    printf ("Rzomiar pamieci zarezerwowanej dla struktury8: %d \n", sizeof (struct struktura8));

    /*.....................................................................
     * i jeszcze sposoby dopelniania rezerwowanej pamieci zalezne od ustawienia zmiennych
     */

    struct struktura9 {
                unsigned char        b; //1B
                unsigned int         a; //4B
                unsigned char        c; //1B
            };
    printf ("Wielkosc pamieci zarezerwowanej dla struktury9: %d \n", sizeof (struct struktura9));

    struct struktura10 {
                    unsigned int         a; //4B
                    unsigned char        b; //1B
                    unsigned char        c; //1B
                };
    printf ("Wielkosc pamieci zarezerwowanej dla struktury10: %d \n", sizeof (struct struktura10));

    struct struktura11 {
                    unsigned char        b; //1B
                    unsigned char        c; //1B
                    unsigned int         a; //4B
                };
    printf ("Wielkosc pamieci zarezerwowanej dla struktury11: %d \n", sizeof (struct struktura11));


    return EXIT_SUCCESS;
}
```

Program W2

```
/*
 ============================================================================
 Name        : W2.c
 Author      : Jaroslaw Krygier, Wojskowa Akademia Techniczna, WEL
 Version     : v1
 Copyright   : Kopiowanie i wykorzystywanie bez zgody wlasciciela zabronione. Mozna uzywac jako referencja
jedynie w ramach zajec.
 Description : Przedmiot: Języki C/C++ W zastosowaniach sieciowych - W2
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // do obslugi funkcji POSIX: sleep

#define INTERWAL 1

int main(void) {

    /************************************
     *Instrukcje sterujace
     ************************************/
     //if, switch, for, while, do while, brak, continue
    printf ("=============Instrukcje sterujace============\n");
    ////////////////////////////////////////////////
    //instrukcja if
    //deklaracja
    int i=1;
    //koniec deklaracji

    //przyklad 1 - warunek
    if (i==1){
        printf ("i=1\n");
    }
    else if (i>1){
        printf ("i>1\n");
    }
    else if (i<1){
        printf ("i<1\n");
    }
    else {
        printf ("i=inna wartosc\n");
    }

    //przyklad 2 - wlaczenie i wylaczenie czesci programu
    if (1){
        printf ("Prawda\n");
    }

    if (0) {
        printf ("Falsz\n");
    }
    ////////////////////////////////////////////////////////
    //instrukcja switch

    switch (i){
        case 0:
            printf ("case: 0 \n");
            break;
        case 1:
            printf ("case: 1 \n");
            break;
        default:
            printf ("case: default \n");
            break;
    }

    ////////////////////////////////////////////////////////
```

```
//instrukcja while : petla wykonwane do czasu gdy prawdziewy jest warunek: warunek sprawdzany jest na
poczatku petli

//>>>>>>>>>>>>>>>>>Przyklad 1 - typowe zastosowanie
i=0;
while (i<5) {
    printf ("while i=%d\n", i);
    i++;
}

//>>>>>>>>>>>>>>>>>Przyklad 2 - petla nieskonczona - wykonanie programu w sposob ciagly

i=0;
while (1){
    printf("petla nieskonczona 'while' i=%d \n", i);
    sleep(INTERWAL);  //uwaga to nie jest funkcja standardowa jezyka C (POSIX)
    i++;
    if (i>10){
        break;
    }
}

////////////////////////////////////////////////////
//instrukcja do {} while () : petla wykonwane do czasu gdy prawdziwy jest warunek: warunek sprawdzany
jest na koncu petli

//>>>>>>>>>>>>>>>>>Przyklad 1 - typowe zastosowanie
i=0;
do {
    printf ("do while i=%d\n", i);
    ++i;
} while (i<5);

//>>>>>>>>>>>>>>>>>Przyklad 2 - petla nieskonczona - wykonanie programu w sposob ciagly
i=0;
do {
    printf("petla nieskonczona 'do while' i=%d \n", i);
    sleep(INTERWAL);  //uwaga to nie jest funkcja standardowa jezyka C (POSIX)
    ++i;
    if (i>10){
        break;
    }
} while (1);

////////////////////////////////////////////////////
//instrukcja for
/*
 for (wyrażenie1; wyrażenie2; wyrażenie3) {
   //instrukcje do wykonania w pętli
 }

.....................
jest rownoznaczne z
.....................

 wyrażenie1;
 while (wyrażenie2) {
     //instrukcje do wykonania w pętli
     wyrażenie3;
 }
*/

//>>>>>>>>>>>>>>>>>Przyklad 1 - typowe zastosowanie
for (i=0;i<5;++i) {
    printf ("for i=%d\n", i);
}

//>>>>>>>>>>>>>>>>>Przyklad 2 - petla nieskonczona - wykonanie programu w sposob ciagly
i=0;
for (;;){
    printf("petla nieskonczona 'for' i=%d \n", i);
```

```
        sleep(INTERWAL);  //uwaga to nie jest funkcja jezyka C (POSIX)
        i++;
        if (i>10){
            break;
        }
    }

    ////////////////////////////////////////////
    //instrukcja continue

    i=0;
    for(;;){
        ++i;
        printf ("testowanie continue, i=%d \n",i);
        sleep(INTERWAL);  //uwaga to nie jest funkcja jezyka C (POSIX)
        if (i>10)
            break;
        if (i>5)
            continue;
        printf ("Gdy i > 5 to ta linia nie bedzie wyswietlana\n");
    }

    ////////////////////////////////////////////
    //goto etykieta - starac sie nie uzywac tej instrukcji: zaciemnia program i utrudnia analize kodu
    i=0;
    for (;;){
        ++i;
        printf ("testowanie goto, i=%d \n", i);
        sleep(INTERWAL);
        if (i>3)
            goto etykieta;
    }
    etykieta:
    printf ("goto: program przeskoczyl do etykiety \n");

    ////////////////////////////////////////////
    // exit ("kod wyjscia") - koniec programu
    i=0;
    for (;;){
        ++i;
        printf ("testowanie exit, i=%d \n", i);
        sleep(INTERWAL);
        if (i>3){
            printf ("Teraz program zakonczy dzialanie \n");
            exit (0); //kod 0 oznacza poprawne zakonczenie programu
        }
    }
    printf ("Tu nie bede \n");
    return EXIT_SUCCESS; // nstrukcja exit zakonczyla program wczesniej
}
```

Program W3

```
/*
 ============================================================================
 Name        : W3.c
 Author      : Jaroslaw Krygier, Wojskowa Akademia Techniczna, WEL
 Version     : v1
 Copyright   : Kopiowanie i wykorzystywanie bez zgody wlasciciela zabronione. Mozna uzywac jako referencja
jedynie w ramach zajec.
 Description : Przedmiot: Języki C/C++ W zastosowaniach sieciowych - W3
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>

//deklaracja funkcji
int funkcja (int, int);
int funkcja_mnozenie ();

int main(void) {
    puts("Wskazniki, operacje na pamieci");

    /*************************************
    * Deklaracje zminnych wskaznikowych
    *************************************/
    printf ("=============Deklaracje zminnych wskaznikowych============\n");
    //deklaracje
    int *ptr_int;
    char    *ptr_char;
    double  *ptr_double;

    int liczba_1=1;
    charznak_1=0xaa;
    double  liczba_2=10.1;
    unsigned char   tablica[6]= {1,};
    // koniec deklaracji

    //Rozmiar zmiennych wskaznikowych
    printf ("Rozmiar zmiennych wskaznikowych (int, char, double): %d, %d, %d \n", sizeof (ptr_int),
    sizeof(ptr_char), sizeof(ptr_double));

    //uwaga na taka deklaracje
    char *a,b,c; // tylko a jest wskaznikiem (4B), b i c sa zmiennymi jednobajtowymi

    //uzyskanie wskaznika do zadeklarowanej zmiennej
    ptr_int = &liczba_1;
    ptr_char = &znak_1;
    ptr_double = &liczba_2;

    //operacje na wskaznikach
    //dostep do danych zaadresownych wskaznikiem
    *ptr_int = 2;
    *ptr_char = 0xbb;
    *ptr_double = 20.0;

    //nazwa tablicy jest wskaznikiem
    *(tablica +1) = 0xcc;
    if ((&tablica[2]) == (tablica+2)) {
        printf ("Dwa sposoby dostepu do adresu komorki w tablicy \n");
        }
    else {
        printf ("Nieoczekiwana reakcja \n");
    }

    //wykorzytanie NULL we wskaznikach
    ptr_int = NULL; //tak przypisany wskaznik nie wskazuje na zadna komorke pamieci
```

```
    /************************************
    * Wskazniki i struktury
    ************************************/
    printf ("=============Wskazniki i struktury============\n");

    //deklaracja
    struct struktura_1 {
        unsigned char   a;
        unsigned int b;
    };
    struct struktura_1 zmienna1;
    struct struktura_1 *struct_ptr;
    //koniec deklaracji

    struct_ptr = &zmienna1;
    //dostep do danych w strukturze
    struct_ptr->a = 0x01;
    struct_ptr->b = 10;
    printf ("Pierwszy element struktury: %x, drugi element struktury: %d \n", struct_ptr->a, struct_ptr->b);

    /*********************************************
    * Dynamiczna alokacja pamieci                                *
    *********************************************/
    printf ("=============Dynamiczna alokacja pamieci============\n");
    //wykorzystanie funkcji malloc i free
    //tablica jednowymiarowa
    //deklaracja
    int *ptr; //wskaznik do tablicy (bufora) zawierajacej liczby typu int
    int wielkosc_tablicy = 10;
    int i;

    ptr = (int*) malloc (wielkosc_tablicy*sizeof(int)); //przydzial pamieci dla tablicy dopiero tutaj
    for (i=0;i<wielkosc_tablicy;++i) {
        ptr[i]=i;
        printf ("%d, ", ptr[i]);
    }
    printf ("\n");
    free (ptr); //zwolnienie pamieci


    /*********************************************
    *   Wskazniki na funkcje                      *
    *********************************************/
    printf ("=============Wskaźniki na funkcje============\n");
    //deklaracja wskaznika na funkcje
    int (*wsk_funkcja)(int a, int b);
    //koniec deklaracji wskaznika
    int wynik_funkcji;

    wsk_funkcja = funkcja; //informujemy jakiej funkcji dotyczy adres

    wynik_funkcji = wsk_funkcja (1,2);
    printf ("wynik funkcji: %d \n", wynik_funkcji);


    return EXIT_SUCCESS;
}

int funkcja (int a, int b){
    int c;
    c=a+b;
    return(c);
}

int funkcja_mnozenie (int a, int b) {
    return a*b;
}
```

Program W4

```
/*
 ============================================================================
 Name        : W4.c
 Author      : Jaroslaw Krygier, Wojskowa Akademia Techniczna, WEL
 Version     : v1
 Copyright   : Kopiowanie i wykorzystywanie bez zgody wlasciciela zabronione. Mozna uzywac jako referencja
jedynie w ramach zajec.
 Description : Przedmiot: Języki C/C++ W zastosowaniach sieciowych - W4
 ============================================================================
 */
/* KOMUNIKACJA Z FUNKCJAMI POPRZEZ ARGUMENTY TYPU WSKAZNIK */
/* WARTOŚCI ZWRACANE PRZEZ FUNKCJE */
/* LISTY WIĄZANE - BUFORY DYNAMICZNE */

#include <stdio.h>
#include <stdlib.h>

//definicja struktur do komunikacji z funkcją
    struct dane_wejsciowe {
        unsigned int     wej1;
        unsigned int     wej2;
    };
    struct dane_wyjsciowe {
        unsigned int     wyj1;
        double           wyj2;
    };
//koniec definicji struktur

// definicja struktury elementu listy wiazanej
struct element_bufora {
        struct element_bufora *nastepny;
        struct element_bufora *poprzedni;
        struct element_bufora *pierwszy;
        unsigned char pakiet[10];
    };
//konice definicji

//deklaracja zmiennej globalnej (adresu) pierwszego elementu listy - zakotwiczenia listy
struct element_bufora *pierwszy;

//deklaracja fukncji
int operacje_na_wskaznikach();

int main(void) {
    puts("W4");

    /*********************************************
    *   Komunikacja z funkcją poprzez wskaźniki       *
    *********************************************/
    printf ("=============Komunikacja z funkcją poprzez wskaźniki============\n");

    //Deklaracja
    struct dane_wejsciowe    wej;
    struct dane_wyjsciowe    wyj;

    struct dane_wejsciowe *wsk_wej;
    struct dane_wyjsciowe *wsk_wyj;
    int poprawnosc_operacji;
    //Koniec deklaracji

    wsk_wej=&wej;
    wsk_wyj=&wyj;

    wsk_wej->wej1=10;
    wsk_wej->wej2=20;

    wsk_wyj->wyj1=0;
    wsk_wyj->wyj2=0.0;
```

```
printf ("A: wyj.wyj1 = %d, wyj.wyj2 = %f \n", wyj.wyj1,  wyj.wyj2);

poprawnosc_operacji = operacje_na_wskaznikach (wsk_wej, wsk_wyj);

if (poprawnosc_operacji == 0) {
    printf ("Blad operacji\n");
}
else {
    printf ("B: wyj.wyj1 = %d, wyj.wyj2 = %f \n", wyj.wyj1,  wyj.wyj2);
}


/*********************************************
*   Zalegle typy zlozone                     *
*   Struktury juz byly.....                      *
*   Typ wyliczeniowy: enum                       *
*   Unie: union                              *
*********************************************/
printf ("=============Typy zlozone============\n");

//wyliczanie
//definicja stalych wyloczanych
enum zbior_opcji {
    opcja_0=3,
    opcja_1,
    opcja_2,
    opcja_3
};

//deklaracja zmiennych
int A, B, C, D;

A = opcja_0;
B = opcja_1;
C = opcja_2;
D = opcja_3;

printf ("A=%d, B=%d, C=%d, D=%d \n", A,B,C,D);

//unie
//definicja unii
union unia {
    unsigned int zmienna_int;
    unsigned char    zmienna_char;
    unsigned short   zmienna_short;
};

//wielkosc pamieci rezerwowanej
printf ("wielkosc pamieci unii: %d \n", sizeof (union unia));

//dostep do zmiennych w unii
//deklaracja
union unia zmienna_unia;
//zapisanie zamiennej
zmienna_unia.zmienna_int = 0xaabbccdd;
//dostep
printf ("zmienna_unia.zmienna_int: %x \n", zmienna_unia.zmienna_int);
printf ("zmienna_unia.zmienna_char: %x \n", zmienna_unia.zmienna_char);
printf ("zmienna_unia.zmienna_short: %x \n", zmienna_unia.zmienna_short);

//przyklad praktyczny
//definicja struktury i unii
struct adres_ip_bajty {
    unsigned char    bajt4;
    unsigned char    bajt3;
    unsigned char    bajt2;
    unsigned char    bajt1;
};

union adres_ip {
```

```
    unsigned int        addr;
    struct adres_ip_bajty    addr_bajty;
};

//deklaracja
union adres_ip adres;

adres.addr_bajty.bajt1 = 148;
adres.addr_bajty.bajt2 = 10;
adres.addr_bajty.bajt3 = 7;
adres.addr_bajty.bajt4 = 255;

printf ("Adres ip w postaci 32 bitow: %x \n", adres.addr);

/*********************************************
*   Pola bitowe  - flagi                     *
*                                            *
*********************************************/
printf ("=============Pola bitowe - sposob zarzadzania flagami============\n");

//definicja struktury
struct flagi {
    unsigned char    zarezerwowane:3, //3 bity
                     f5:1,               //1 bit
                     f4:1,               //1 bit
                     f3:1,               //1 bit
                     f2:1,               //1 bit
                     f1:1;               //1 bit

};

printf ("Rozmiar struktury 'flagi':%d B\n", sizeof (struct flagi));

//deklaracja zmiennej
struct flagi     pole_flag;
struct flagi     *ptr;
unsigned char    oktet;

//wyczyszczenie pola flag oraz zmiennej octet
ptr = &pole_flag;
memset (ptr,0,1); // wstawienie 0 na 1 bajcie
memset (&oktet,0,1);

//ustawienie flag
pole_flag.f1 = 1;
pole_flag.f2 = 1;
pole_flag.f3 = 0;
pole_flag.f4 = 1;
pole_flag.f5 = 0;
pole_flag.zarezerwowane = 111;

memcpy (&oktet,ptr,1); //sprawdzic w debugerze zapis bitowy zmiennej 'octet'


/*********************************************
*   Listy wiazane - bufory dynamiczne         *
*                                             *
*********************************************/
printf ("=============Listy wiazane============\n");
//deklaracja zmiennej
int i;
struct element_bufora *nowy_element;
struct element_bufora *poprzedni_element;

//obserwowac w trybie debugera

//przydzial pamieci dla pierwszego elementu
pierwszy = (struct element_bufora*) malloc (sizeof(struct element_bufora));
pierwszy->nastepny = NULL;
pierwszy->poprzedni = NULL;
pierwszy->pierwszy = pierwszy;
```

```
    memset (pierwszy->pakiet, 0, 10);


    //dodanie pozostalych elementów do listy
    poprzedni_element = pierwszy;
    for (i=0;i<5;i++) {
        nowy_element = (struct element_bufora*) malloc (sizeof (struct element_bufora));
        nowy_element->poprzedni = poprzedni_element;  //zwiazanie z poprzednim elementem
        nowy_element->nastepny = NULL;                //ostatni element nie ma nstepnika
        nowy_element->pierwszy = pierwszy;            //zwiazanie z zakotwiczeniem

        poprzedni_element->nastepny = nowy_element;   //zwiazanie poprzedniego elementu z nowym

        poprzedni_element = nowy_element;             //teraz nowy element jest juz gotowy do przyjecia
                                                      //nastepnika
    }



    return EXIT_SUCCESS;
}

//Zawartosc funkcji
int operacje_na_wskaznikach (struct dane_wejsciowe *wej_ptr, struct dane_wyjsciowe *wyj_ptr){

    int wynik1;
    double  wynik2;

    wynik1 = wej_ptr->wej1+wej_ptr->wej2;
    wynik2 = (double) (wej_ptr->wej1) / (double) (wej_ptr->wej2);

    wyj_ptr->wyj1= wynik1;
    wyj_ptr->wyj2= wynik2;

    //operacja przebiegla pomyslnie
    return (1);
}
```

Program W5

```
/*
 ============================================================================
 Name        : W5.c
 Author      : Jaroslaw Krygier, Wojskowa Akademia Techniczna, WEL
 Version     : v1
 Copyright   : Kopiowanie i wykorzystywanie bez zgody wlasciciela zabronione. Mozna uzywac jako referencja
jedynie w ramach zajec.
 Description : Przedmiot: Języki C/C++ W zastosowaniach sieciowych - W5
 ============================================================================

/* ZAAWANSOWANE OPERACJE NA WSKAŹNIKACH */
/* OPERACJE NA WSKAŹNIKACH DO FUNKCJI - ROZSZERZENIE */


#include <stdio.h>
#include <stdlib.h>

//fukcja dodawnia
int dodawanie(int x, int y) {
    return (x+y);
}

//funkcja odejmowania
int odejmowanie (int x, int y) {
    return (x-y);
}

int operacja (int (*) (int, int), int, int); // definicja ponizej fukcji main()

int main(void) {
    //Zaawansowane operacje na wskaznikach

    int a;
    int b=5;
    int c=6;
    int* wsk;

    ////////////////////
    //wskazniki
    ////////////////////

    wsk = &a;
    printf ("Adres zmiennej 'a': 0x%x\n", (unsigned int) wsk);  // rzutowanie na unsigned int, poniewaz
    printf wymaga takiego typu (nie ma warningow)
    wsk = &b;
    printf ("Adres zmiennej 'b': 0x%x\n", (unsigned int) wsk);
    wsk = &c;
    printf ("Adres zmiennej 'c': 0x%x\n", (unsigned int) wsk);

    ////////////////////
    //wskazniki do funkcji
    ////////////////////
    //deklaracja
    int (*funkcja) (int, int);
    //jaka jest roznica z int *funkcja (int, int);

    funkcja = &dodawanie;
    // teraz wskaznik wskazuje na fukncje 'dodawanie'
    printf ("1) Wskaznik na funkcje 'dodawanie': 0x%x\n",(unsigned int) funkcja);

    //lub alternatywnie
    funkcja = dodawanie;
    // teraz wskaznik wskazuje na fukncje 'dodawanie'
    printf ("2) Wskaznik na funkcje 'dodawanie': 0x%x\n", (unsigned int) funkcja);

    //Wywolanie funkcji przez wskanik
    a = funkcja (b,c);
    printf ("1) Wynik: %d \n", a);
```

```
// lub alternatywnie
a = (*funkcja) (b,c);
printf ("2) Wynik: %d \n", a);

// przypisanie do wskazniaka roznych funkcji
//teraz ten sam wskaznik wskazuje na inna funkcje
funkcja = odejmowanie; //mozna przypisac funkcje bez &
printf ("3) Wskaznik na funkcje 'odejmowanie': 0x%x\n", (unsigned int) funkcja);
//wywolanie jest identyczne, ale wynik rozny
a = (*funkcja) (b,c);
printf ("3) Wynik: %d \n", a);

////////////////////
//tablica wskaznikow do funkcji
////////////////////

// deklaracja tablicy wskaznikow do fukncji
int (*tablica_wskaznikow_do_funkcji [3]) (int, int);

//przypisanie adresow fukcji do komorek tablicy
tablica_wskaznikow_do_funkcji [0] = dodawanie;
tablica_wskaznikow_do_funkcji [1] = odejmowanie;
tablica_wskaznikow_do_funkcji [2] = NULL;

printf ("Wskaznik na funkcje 'dodawanie' [0]: 0x%x\n", (unsigned int) tablica_wskaznikow_do_funkcji[0]);
printf ("Wskaznik na funkcje 'odejmowanie'[1]: 0x%x\n", (unsigned int)
        tablica_wskaznikow_do_funkcji[1]);
printf ("Wskaznik na funkcje 'NULL'[2]: 0x%x\n", (unsigned int) tablica_wskaznikow_do_funkcji [2]);

//wywolanie funkcji z tablicy

a = (*tablica_wskaznikow_do_funkcji [0]) (b,c);
printf ("1) Wynik: %d \n", a);
a = (*tablica_wskaznikow_do_funkcji [1]) (b,c);
printf ("2) Wynik: %d \n", a);

//lub alternatywnie
a = tablica_wskaznikow_do_funkcji [0] (b,c);
printf ("1) Wynik: %d \n", a);
a = tablica_wskaznikow_do_funkcji [1] (b,c);
printf ("2) Wynik: %d \n", a);

/*UWAGA!!!*/
/*
 * Wskaniki do fukcji mozna:
- przekazywac do innych fukcji jako argumenty
- zwracac jako wynik dzialania fukcji
- porownywac z NULL
* Nie mozma
- wykonywac operacji matematycznych
*/

//Przekazanie wskaznika do fukcji jako argumentu innej fukcji
funkcja = dodawanie;
a = operacja (funkcja, 3, 5);
printf ("1) Wynik: %d \n", a);
//lub bezposrednio nazwa fukcji jest tez wskaznikiem
a = operacja (dodawanie, 3, 5);
printf ("2) Wynik: %d \n", a);

// a teraz przekazujemy wskznik innej fukcji
a = operacja (odejmowanie, 3, 5);
printf ("3) Wynik: %d \n", a);

//sprawdzenie
//Napisac fukcje 'operacja_arytm', ktora wykonuje y=(a/b)+c
//Napisac fukcje o nazwie 'operacje_new' z argumentami (typ_operacji, arg1, arg2, arg3) zwracajaca
wynik:
// dodawania lub odejmowania, lub wynik fukcji 'operacja_arytm' w zaleznosci od przekazanego typu
operacji
```

```
    return EXIT_SUCCESS;
}

// fukcja z przekazanym wskaznikiem do fukcji
int operacja (int (*wsk_do_funkcji_przekazany) (int, int), int a, int b){
    // jako pierwszy argument bedzie przekazywany wskaznik do fukcji

    int wynik;

    wynik = wsk_do_funkcji_przekazany (a,b);
    // lub
    //wynik = (*wsk_do_funkcji_przekazany) (a,b);

    return wynik;
}
```