



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

Informatyka 1

Kod przedmiotu: **EMS1A1004**

(studia stacjonarne)

JĘZYK C - INSTRUKCJA WARUNKOWA IF, OPERATORY RELACYJNE I LOGICZNE, WYRAŻENIA LOGICZNE, ZAGNIEŹDŻANIE IF-ELSE, OPERATOR WARUNKOWY, INSTRUKCJA SWITCH

Numer ćwiczenia

INF1_E03

Autor:
dr inż. Jarosław Forenc

Białystok 2025

Spis treści

1. Opis stanowiska	3
1.1. Stosowana aparatura	3
1.2. Oprogramowanie	3
2. Wiadomości teoretyczne.....	3
2.1. Instrukcja warunkowa if	3
2.2. Operatory relacyjne (porównania)	6
2.3. Operatory logiczne	7
2.4. Wyrażenia logiczne	8
2.5. Przykłady obliczania wartości wyrażeń logicznych	8
2.6. Zagnieżdżanie if-else	11
2.7. Operator warunkowy	13
2.8. Instrukcja wyboru wielowariantowego - switch.....	14
2.9. Stałe zadeklarowane jako const	20
3. Przebieg ćwiczenia.....	20
4. Literatura.....	24
5. Pytania kontrolne	24
6. Wymagania BHP	25

Materiały przygotowane w ramach projektu „PB 5.0 - dostosowanie oferty dydaktycznej Politechniki Białostockiej do potrzeb nowoczesnej gospodarki oraz zielonej i cyfrowej transformacji” (nr umowy FERS.01.05-IP.08-0327/23-00) w ramach programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027 współfinansowanego ze środków Europejskiego Funduszu Społecznego Plus.

Publikacja jest udostępniona na licencji Creative Commons - Uznanie autorstwa 4.0 (CC BY 4.0). Pełna treść licencji dostępna na stronie creativecommons.org.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10/11.

1.2. Oprogramowanie

Na komputerach zainstalowany jest edytor kodu źródłowego Visual Studio Code 1.103 (lub nowszy) wraz z odpowiednimi rozszerzeniami (C/C++, Code Runner, Polish Language Pack for Visual Studio Code) oraz MinGW - zestaw kompilatorów różnych języków programowania (m.in. C, C++, Fortran, Java).

2. Wiadomości teoretyczne

2.1. Instrukcja warunkowa if

Instrukcja warunkowa **if** służy do sprawdzania warunków logicznych i decydowaniu o wykonywaniu lub też pomijaniu fragmentów kodu programu. Instrukcja ta może występować w dwóch postaciach.

Postać nr 1 instrukcji **if**:

```
if (wyrażenie)  
    instrukcja1;
```

Jeśli **wyrażenie** w nawiasach jest prawdziwe, to wykonywana jest **instrukcja1**. Gdy **wyrażenie** to jest fałszywe, to **instrukcja1** nie jest wykonywana. Z punktu widzenia języka C i instrukcji **if**, **wyrażenie** jest prawdziwe, gdy jego wartość jest różna od zera. Natomiast **wyrażenie** jest fałszywe, gdy jego wartość jest równa zero.

W poniższym fragmencie programu obliczana jest wartość bezwzględna zmiennej **x**. Jeśli zmienna **x** jest mniejsza od zera, to jej znak jest zmieniany na przeciwny. Jeśli natomiast **x** jest większe lub równe zero, to nic się nie dzieje.

```
if (x < 0)
    x = -x;
```

Wcięcie akapitowe (kilka spacji przed instrukcją `x = -x;`) nie jest wymagane przez składnię języka, ale należy do powszechnej praktyki formatowania kodu programu. Wcięcia tego rodzaju wyróżniają instrukcje, które nie są wykonywane zawsze, ale jedynie w sytuacji spełnienia pewnego warunku.

Postać nr 2 instrukcji `if`:

```
if (wyrażenie)
    instrukcja1;
else
    instrukcja2;
```

Jeśli **wyrażenie** w nawiasach jest prawdziwe, to wykonywana jest **instrukcja1**, natomiast **instrukcja2** wówczas nie jest wykonywana. W przeciwnym przypadku, jeśli **wyrażenie** w nawiasach nie jest prawdziwe, to wykonywana jest **instrukcja2**, a **instrukcja1** jest pomijana.

Poniższy fragment programu sprawdza, czy osoba o podanym wieku jest pełnoletnia.

```
if (wiek >= 18)
    printf("Osoba jest pełnoletnia\n");
else
    printf("Osoba nie jest pełnoletnia\n");
```

Jeśli, w przypadku spełnienia warunku w instrukcji `if`, ma być wykonana więcej niż jedna instrukcja, to należy instrukcje te objąć dodatkowymi nawiasami klamrowymi. Jest to tzw. instrukcja **złożona** (instrukcja **grupująca**, **blok**).

```
if (x > 0)
{
    printf("Liczba jest większa od zera\n");
    printf("Wartosc liczby: %d \n", x);
}
```

Poniższy program oblicza iloraz dwóch liczb. Dzielenie jest wykonywane tylko wtedy, gdy wartość zmiennej **b** jest różna od zera. Jeśli **b** ma wartość zero, to program wyświetla odpowiedni komunikat.

Program obliczający iloraz dwóch liczb wprowadzonych z klawiatury.

```
#include <stdio.h>

int main(void)
{
    float a, b, w;

    printf("Podaj pierwsza liczbe: ");
    scanf("%f",&a);
    printf("Podaj druga liczbe:    ");
    scanf("%f",&b);

    if (b != 0)
    {
        w = a / b;
        printf("Wynik dzielenia to:    %f\n",w);
    }
    else
    {
        printf("Dzielenie przez zero\n");
    }

    return 0;
}
```

Przykładowe wyniki uruchomienia programu:

```
Podaj pierwsza liczbe: 5
Podaj druga liczbe:    2
Wynik dzielenia to:    2.500000
```

```
Podaj pierwsza liczbe: 5
Podaj druga liczbe:    0
Dzielenie przez zero
```

```
Podaj pierwsza liczbe: -2.5
Podaj druga liczbe:    3
Wynik dzielenia to:    -0.833333
```

W powyższym programie **instrukcja złożona** występuje dwukrotnie - po **if** i po **else**. W tym drugim przypadku nie jest to konieczne, gdyż mamy tylko jedną instrukcję. Całą instrukcję **if** można zatem zapisać także w następujący sposób:

```
if (b != 0)
{
    w = a / b;
    printf("Wynik dzielenia to: %f\n",w);
}
else
    printf("Dzielenie przez zero\n");
```

Wyrażenie występujące w instrukcji **if** musi być zawsze umieszczone w nawiasach zwykłych. Po nawiasie nie stawia się średnika. Konstrukcja ze średnikiem na końcu:

```
if (wyrażenie) ;
    instrukcja1;
```

jest poprawna (kompilator nie zasygnalizuje błędu), ale oznacza wykonanie **instrukcji pustej**, jeśli **wyrażenie** jest prawdziwe. Natomiast **instrukcja1** zostanie wykonana zawsze, niezależnie od tego czy **wyrażenie** jest prawdziwe, czy też nie.

Jako **wyrażenie** w instrukcji **if** najczęściej stosowane jest **wyrażenie logiczne**. Wyrażenie takie może zawierać nazwy zmiennych, stałe liczbowe, operatory relacyjne (porównania), operatory logiczne, operatory arytmetyczne i dodatkowe nawiasy zwykłe. Operatory relacyjne i logiczne oraz sposób tworzenia i obliczania wartości wyrażeń logicznych opisano w dalszej części instrukcji.

2.2. Operatory relacyjne (porównania)

Operatory relacyjne sprawdzają prawdziwość zadanych za ich pomocą warunków logicznych. Wynik takiego porównania jest wartością typu **int** i jest równy:

- 1 - gdy warunek jest prawdziwy;
- 0 - gdy warunek nie jest prawdziwy (fałszywy).

Operatory relacyjne występujące w języku C zestawiono w Tabeli 1.

Tabela 1. Operatory relacyjne (porównania) w języku C

Operator	Przykład	Znaczenie
>	a > b	a większe od b
<	a < b	a mniejsze od b
>=	a >= b	a większe lub równe b
<=	a <= b	a mniejsze lub równe b
==	a == b	a równe b
!=	a != b	a nierówne b (a różne od b)

2.3. Operatory logiczne

W języku C występują trzy operatory logiczne, które zostały zestawione w Tabeli 2.

Tabela 2. Operatory logiczne w języku C

Operator	Znaczenie	Opis
!	NOT, nie	jednoargumentowy operator negacji logicznej - zmienia argument różny od zera na wartość 0 , a argument równy zero na wartość 1
&&	AND, i	dwuargumentowy operator koniunkcji, iloczyn logiczny
	OR, lub	dwuargumentowy operator alternatywy, suma logiczna

Zasadę działania poszczególnych operatorów przedstawiają Tabele 3 i 4.

Tabela 3. Operator negacji

a	! a
falsz	prawda
prawda	falsz

Tabela 4. Operatory koniunkcji i alternatywy

a	b	a && b	a b
falsz	falsz	falsz	falsz
falsz	prawda	falsz	prawda
prawda	falsz	falsz	prawda
prawda	prawda	prawda	prawda

2.4. Wyrażenia logiczne

Z operatorów relacyjnych (porównania) oraz operatorów logicznych budowane są wyrażenia logiczne. W wyrażeniach logicznych mogą występować również zmienne, stałe liczbowe, operatory arytmetyczne, operator przypisania i wywołania funkcji zwracającej wynik. Podczas obliczania wartości wyrażenia logicznego uwzględniany jest priorytet operatorów przedstawiony w Tabeli 5.

Tabela 5. Priorytet wybranych operatorów (od najwyższego do najniższego)

Operator	Typ operatora
!	logiczny
* / %	arytmetyczne
+ -	arytmetyczne
> < >= <=	relacyjne
== !=	relacyjne
&&	logiczny
 	logiczny
=	przypisania

2.5. Przykłady obliczania wartości wyrażeń logicznych

Wyrażenia logiczne obliczane są od strony lewej do prawej. Proces obliczeń kończy się już w momencie, gdy tylko wiadomo, jaki będzie wynik całego wyrażenia. Załóżmy, że mamy w programie następujące deklaracje zmiennych:

```
int i = 1;
int j = 2;
int k = -5;
```

Wyrażenie	Wartość
<code>i == 1</code>	1 (prawda)

Sprawdzamy, czy zmienna `i` jest równa `1`. Ponieważ jest to prawda, to wartością całego wyrażenia jest `1`. Uwaga: należy zwrócić szczególną uwagę na wprowadzenie operatora porównania (dwa znaki równości), a nie operatora przypisania (jeden znak równości).

Wyrażenie	Wartość
<code>j = 3</code>	3 (prawda)

Sprawdzamy, czy zmienna `j` ma wartość `3`. Przez pomyłkę zamiast dwóch znaków równości wprowadzamy tylko jeden. Wówczas zmiennej `j` zostaje przypisana nowa wartość - `3`, więc wartością całego wyrażenia jest `3`. Jeśli wyrażenie takie pojawi się w instrukcji warunkowej `if`, to okaże się, że jest ono prawdziwe!!! W języku C, w instrukcji `if`, wyrażenie jest prawdziwe, gdy jego wartość jest różna od zera. Niektóre kompilatory wyświetlają ostrzeżenie po napotkaniu operatora przypisania w instrukcji `if`.

Wyrażenie	Wartość
<code>j != 3</code>	1 (prawda)

Sprawdzamy, czy wartość zmiennej `j` jest różna od `3` (nie jest równa `3`). Ponieważ `j` jest równe `2`, to wartością wyrażenia logicznego jest `1` czyli prawda.

Wyrażenie	Wartość
<code>j =! 3</code>	0 (fałsz)

Przy sprawdzaniu, czy wartość zmiennej **j** jest różna od **3**, przez pomyłkę zapisujemy odwrotnie symbole tworzące operator. Kompilator nie sygnalizuje żadnego błędu. W rzeczywistości w wyrażeniu tym stosujemy dwa operatory: przypisania (=) oraz negacji logicznej (!). Wyższy priorytet ma negacja. Wartość różna od zera (**3**) zostanie zamieniona na zero (**!3 = 0**), które następnie zostanie przypisane zmiennej **j**. Wartość zmiennej **j** będzie wartością całego wyrażenia logicznego (0 czyli fałsz).

Wyrażenie	Wartość
i + j < k	0 (fałsz)

W wyrażeniach logicznych mogą być również stosowane operatory arytmetyczne. W powyższym przykładzie jako pierwsze zostanie wykonane dodawanie (operator **+** ma wyższy priorytet niż operator **<**), a następnie wynik dodawania zostanie porównany z wartością zmiennej **k**.

Wyrażenie	Wartość
3 < j < 6	1 (prawda)

Powyższy przykład pokazuje jeden z najczęstszych błędów popełnianych przez początkujących programistów. Chcemy sprawdzić, czy zmienna **j** $\in$ (**3,6**). Jako pierwsze wykonywane jest porównanie **3 < j**. Ponieważ nie jest to prawda (gdyż **j = 2**), to wynikiem porównania jest wartość **0**. Następnie wynik tego porównania (czyli **0**, a nie **j!!!**) jest porównywany z wartością **6** (**0 < 6**). Wyrażenie to jest prawdziwe, a zatem wynik całego wyrażenia logicznego to prawda, czyli **1**. W rzeczywistości jednak **j** $\notin$ (**3,6**). Prawidłowy zapis warunku logicznego sprawdzającego, czy **j** $\in$ (**3,6**) przedstawiony jest poniżej.

Wyrażenie	Wartość
j > 3 && j < 6	0 (fałsz)

Jako pierwsze jest obliczane wyrażenie po lewej stronie: **j > 3**. Wynikiem tego porównania jest **0** (**fałsz**). W tym momencie zakończy się analiza wyrażenia, gdyż

niezależnie od tego co zostanie otrzymane po prawej stronie operatora **&&**, to i tak wartość całego wyrażenia będzie równa **0 (fałsz)**.

Wyrażenie	Wartość
<code>(j >= 0 && j <= 4) (j > 6 && j < 10)</code>	1 (prawda)

Przy obliczaniu wartości powyższego wyrażenia występuje podobna sytuacja jak poprzednio. Wyrażenia `j >= 0` oraz `j <= 4` są prawdziwe, a zatem wyrażenie po prawej stronie operatora `||` nie będzie już obliczane, gdyż całkowity wynik jest już znany. Operator alternatywy logicznej `||` ma niższy priorytet niż operator koniunkcji `&&`, w związku z tym można pominąć nawiasy zwykłe.

W przypadku sprawdzania czy zmienna lub wyrażenie jest równe lub różne od zera można w instrukcji warunkowej **if** zastosować skrócony zapis.

<code>if (j == 0) instrukcja;</code>	można zastąpić przez:	<code>if (!j) instrukcja;</code>
<code>if (j != 0) instrukcja;</code>	można zastąpić przez:	<code>if (j) instrukcja;</code>

2.6. Zagnieżdżanie if-else

Jako instrukcja po **if** może występować kolejny **if** zawierający **else**. Do której instrukcji **if** zatem on należy? Ogólna zasada: danemu **else** odpowiada pierwszy poprzedzający go i znajdujący się w tym samym bloku **if** nie mający jeszcze swojej „pary” w postaci **else**.

W poniższym przykładzie **else** przyporządkowany jest do **if** (**wyrażenie2**):

```
if (wyrażenie1)
    if (wyrażenie2)
        instrukcja1;
    else
        instrukcja2;
```

Przykład:

```
if (delta >= 0)
    if (delta > 0)
        printf("Dwa pierwiastki\n");
    else
        printf("Jeden podwójny pierwiastek \n");
```

Stosując dodatkowe nawiasy klamrowe można przyporządkować **else** do pierwszej instrukcji **if**: **if (wyrażenie1)**:

```
if (wyrażenie1)
{
    if (wyrażenie2)
        instrukcja1;
}
else
    instrukcja2;
```

Standard języka C pozwala na obsługę co najmniej 127 poziomów zagnieżdżania **if-else**:

```
if (wyrażenie1)
    instrukcja1;
else
    if (wyrażenie2)
        instrukcja2;
    else
        if (wyrażenie3)
            instrukcja3;
        else
            if (wyrażenie4)
                instrukcja5;
            else
                if (wyrażenie5)
                    instrukcja6;
                else
                    ...
```

2.7. Operator warunkowy

Operator warunkowy składa się z dwóch symboli (**?** - znak zapytania, **:** - dwukropek) i wymaga trzech operandów (wyrażeń **w1**, **w2**, **w3**). Stosując operator warunkowy, otrzymujemy następującą postać wyrażenia:

w1 ? w2 : w3

Wyrażenie warunkowe obliczane jest w następujący sposób: najpierw obliczane jest wyrażenie **w1**. Jeśli jego wartość jest różna od zera, to obliczane jest wyrażenie **w2** a jego wartość staje się wartością całego wyrażenia warunkowego. W przeciwnym razie **w2** jest ignorowane, a wartością wyrażenia warunkowego staje się wartość wyrażenia **w3** (po wcześniejszym jego obliczeniu).

Wyrażenia warunkowe stosowane są najczęściej wtedy, gdy pewnej zmiennej nadawana jest jedna z dwóch możliwych wartości. Mogą one zastępować proste instrukcje **if ... else**. Przykładowo, obliczanie wartości bezwzględnej zmiennej **x** może mieć następującą postać:

```
if (x < 0)
    y = -x;
else
    y = x;
```

można zastąpić
przez:

```
y = (x < 0) ? -x : x;
```

Podobnie postępujemy z wyznaczeniem większej z dwóch zmiennych **a** i **b**:

```
if (a > b)
    max = a;
else
    max = b;
```

można zastąpić
przez:

```
max = (a > b) ? a : b;
```

Zastosowanie wyrażania warunkowego upraszcza kod programu i może skutkować bardziej zoptymalizowanym kodem wykonywalnym podczas kompilacji. W poniższym przykładzie operator warunkowy został użyty bezpośrednio w instrukcji **printf()** do sprawdzenia czy liczba jest parzysta czy nieparzysta oraz dodatnia czy ujemna.

Sprawdzenie czy liczba jest parzysta/nieparzysta, dodatnia/ujemna.

```
#include <stdio.h>

int main(void)
{
    int x;

    printf("Podaj liczbę: ");
    scanf("%d",&x);

    if (x == 0)
        printf("Liczba: zero\n");
    else
    {
        printf("Liczba: %s\n", x>0 ? "dodatnia" : "ujemna");
        printf("Liczba: %s\n", x%2==0 ? "parzysta" :
            "nieparzysta");
    }
    return 0;
}
```

Przykładowe wyniki uruchomienia programu:

```
Podaj liczbę: 5
Liczba: dodatnia
Liczba: nieparzysta
```

```
Podaj liczbę: -6
Liczba: ujemna
Liczba: parzysta
```

```
Podaj liczbę: 0
Liczba: zero
```

2.8. Instrukcja wyboru wielowariantowego - switch

Instrukcja **switch** służy do podejmowania decyzji wielowariantowych. W instrukcji tej sprawdza się, czy wartość pewnego wyrażenia pasuje do jednej z kilku **całkowitych, stałych wartości** (**wyrażenie stałe**). W przypadku stwierdzenia równości następuje przekazanie sterowania (skok) do odpowiedniego

miejsca. W niektórych sytuacjach instrukcja **switch** może zastąpić wielokrotne instrukcje **if - else if**.

Ogólna postać instrukcji **switch** jest następująca:

```
switch (wyrażenie)
{
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    ...
    default: instrukcje;
}
```

Obliczane jest **wyrażenie** w nawiasach. Następnie jego wartość porównywana jest z wartościami **wyrażeń stałych** (zawartych w częściach oznaczanych przez etykiety **case**). Sterowanie jest przekazywane do tej instrukcji, którą poprzedza etykieta **case** z **wyrażeniem stałym** równym co do wartości **wyrażeniu** w nawiasach **switch**. Od tego miejsca wykonywane są wszystkie instrukcje znajdujące się po tej etykiecie oraz oznaczone przez inne etykiety, aż do końca instrukcji **switch**. Jeśli nie znajdzie się żadna etykieta „pasująca” do wartości **wyrażenia** w nawiasach **switch**, to sterowanie jest przekazywane do części oznaczonej przez etykietę **default**. Jeśli nie ma etykiety **default**, to sterowanie przekazywane jest do instrukcji następującej po **switch**. Etykieta **default** może wystąpić tylko jeden raz. W składni instrukcji **switch** wszystkie instrukcje oraz etykieta **default** są opcjonalne.

Wyrażania stałe występujące po etykietach **case** nie mogą powtarzać się. Jeśli wystąpi taka sytuacja, to kompilator zasygnalizuje błąd. W jednej instrukcji **switch** może występować maksymalnie do 1023 etykiet **case**. **Wyrażenie stałe** musi mieć typ całkowity. Jego wartość powinna być znana w trakcie kompilacji i nie może zostać zmieniona w fazie wykonania programu. Jako **wyrażenie stałe** najczęściej stosuje się:

- liczby całkowite, np. **1, 2, 3, 0, -1, -2**;
- stałe zadeklarowane jako **const** lub przez dyrektywę preprocesora **#define**;
- znaki umieszczone w apostrofach, np. **'+', 'a'**.

W poniższym programie funkcja **getchar()** odczytuje wciśnięty klawisz i przypisuje jego kod do zmiennej **key**. Następnie w instrukcji **switch** kod klawisza jest porównywany z wyrażeniami stałymi znajdującymi się po **case**. Jeśli wciśniętym klawiszem był znak '+', to zmienne **x1** i **x2** są dodawane do siebie, a ich suma jest wyświetlana. Jeśli wciśnięto znak '-', to zmienne są odejmowane. Wprowadzenie innego znaku spowoduje wyświetlenie tekstu: **Nieznana operacja!**.

Wybór arytmetycznego działania w zależności od wciśniętego klawisza.

```
#include <stdio.h>

int main(void)
{
    int    key;
    float  x1 = 10.0, x2 = 5.0, y;

    printf("Podaj dzialanie (+,-): ");
    key = getchar();

    switch (key)
    {
        case '+':
            y = x1 + x2;
            printf("Dodawanie: y = %.2f \n",y);
            break;
        case '-':
            y = x1 - x2;
            printf("Odejmowanie: y = %.2f \n",y);
            break;
        default:
            printf("Nieznana operacja!\n");
    }

    return 0;
}
```

Przykładowe wyniki uruchomienia programu:

```
Podaj dzialanie: +
Dodawanie: y = 15.00
```

Podaj dzialanie: -
Odejmowanie: y = 5.00

Podaj dzialanie: *
Nieznana operacja!

Po instrukcjach każdego wariantu **case** występują instrukcje **break**. Powodują one natychmiastowe opuszczenie instrukcji **switch**. Ich brak spowodowałby wykonanie wszystkich instrukcji (do końca instrukcji **switch**) występujących po każdym **case**.

```
switch (key)
{
case '+':
    y = x1 + x2;
    printf("Dodawanie: y = %.2f \n",y);
case '-':
    y = x1 - x2;
    printf("Odejmowanie: y = %.2f \n",y);
default:
    printf("Nieznana operacja!\n");
}
```

Po wciśnięciu '+' wyświetlone zostałyby komunikaty:

Dodawanie: y = 15.00
Odejmowanie: y = 5.00
Nieznana operacja!

Po wciśnięciu '-' wyświetlone zostałyby komunikaty:

Odejmowanie: y = 5.00
Nieznana operacja!

Po wciśnięciu innego znaku wyświetlony zostałby komunikat:

Nieznana operacja!

Kolejny program wyświetla słownie ocenę wczytaną z klawiatury.

Program wyświetlający słownie ocenę wprowadzoną jako liczba.

```
#include <stdio.h>

int main(void)
{
    int ocena;

    printf("Podaj ocene: ");
    scanf("%d",&ocena);
    switch (ocena)
    {
        case 5:
            printf("Twoja ocena: bardzo dobry\n");
            break;
        case 4:
            printf("Twoja ocena: dobry\n");
            break;
        case 3:
            printf("Twoja ocena: dostateczny\n");
            break;

        case 2:
            printf("Twoja ocena: niedostateczny\n");
            break;
        default:
            printf("Bledna ocena\n");
    }

    return 0;
}
```

Przykłady uruchomienia programu:

```
Podaj ocene: 4
Twoja ocena: dobry
```

```
Podaj ocene: 2
Twoja ocena: niedostateczny
```

```
Podaj ocene: 0
Bledna ocena
```

Z każdym wariantem może być związane jedno lub kilka wyrażeń stałych.

Program wyświetlający informację o wprowadzonej ocenie.

```
#include <stdio.h>

int main(void)
{
    int ocena;

    printf("Podaj ocene: ");
    scanf("%d",&ocena);
    switch (ocena)
    {
        case 5: case 4: case 3:
            printf("Ocena pozytywna\n");
            break;
        case 2:
            printf("Ocena negatywna\n");
            break;

        default:
            printf("Bledna ocena\n");
    }

    return 0;
}
```

Przykłady uruchomienia programu:

Podaj ocene: 4
Ocena pozytywna

Podaj ocene: 3
Ocena pozytywna

Podaj ocene: 2
Ocena negatywna

Podaj ocene: 0
Bledna ocena

2.9. Stałe zadeklarowane jako const

Stałą specyfikuje się wstawiając przed nazwą typu słowo **const**. Tak zadeklarowana “zmienna” musi otrzymać wartość w trakcie inicjalizacji i nie może być później zmieniana.

```
const int x = 10;
```

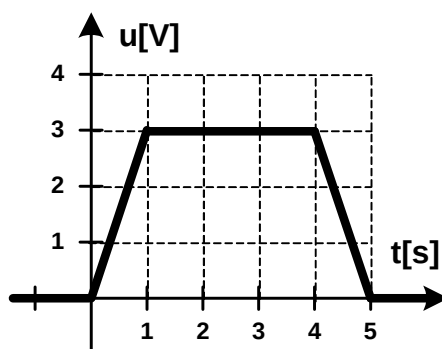
3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Minimalna niebezpieczna dla człowieka wartość prądu przemiennego płynącego przez jego ciało przez dłuższy czas wynosi 0,03 A (30 mA). Napisz program, w którym użytkownik wprowadza z klawiatury prąd w amperach. Program powinien wyświetlić informację, czy wprowadzona wartość prądu jest bezpieczna czy niebezpieczna.
2. Napisz program, w którym użytkownik wprowadza z klawiatury moc odbiornika w watach, a program wyświetla informację o przypisaniu odbiornika do jednej z trzech grup:
 - poniżej 50 W - odbiornik małej mocy;
 - od 50 W do 2000 W - odbiornik średniej mocy;
 - powyżej 2000 W - odbiornik dużej mocy.
3. Napisz program, który wczytuje trzy liczby rzeczywiste, będące wynikami pomiarów napięcia w poszczególnych fazach sieci zasilającej. Wyświetl wartość największego oraz najmniejszego napięcia.
4. Napisz program, który wczytuje trzy liczby rzeczywiste, będące wynikami pomiarów napięcia w poszczególnych fazach sieci zasilającej. Wyświetl

wartości napięć od najmniejszej do największej, a następnie od największej do najmniejszej.

5. Napisz program, w którym użytkownik wprowadza z klawiatury liczbę całkowitą. Program powinien sprawdzić i wyświetlić informację o parzystości (parzysta/nieparzysta) i znaku liczby (dodatnia/ujemna). W przypadku wprowadzenia zera na ekranie powinien wyświetlić się tylko komunikat: zero.
6. Napisz program, w którym wczytywane są trzy liczby: dolna i górna granica pewnego przedziału częstotliwości sygnału elektrycznego oraz dowolna liczba x . Jeśli dolna granica jest większa od górnej, to program powinien wyświetlić komunikat błędu i zakończyć działanie. W przeciwnym przypadku, program powinien wyświetlić informację, czy x :
 - a) znajduje się poniżej przedziału;
 - b) jest dolną granicą przedziału;
 - c) znajduje się w przedziale (ale nie jest jego granicą);
 - d) jest górną granicą przedziału;
 - e) znajduje się powyżej przedziału.
7. Na Rys. 1 przedstawiony jest przebieg napięciowego impulsu trapezowego. Napisz program, który na podstawie wczytanego z klawiatury czasu t (wartość rzeczywista) obliczy i wyświetli odpowiadającą mu wartość napięcia u .



Rys. 1. Przebieg impulsu trapezowego

8. Napisz program rozwiązujący równanie kwadratowe:

$$ax^2 + bx + c = 0 \quad (1)$$

Współczynniki **a**, **b**, **c** wczytaj z klawiatury. Jeśli z wprowadzonych danych wynika, że nie jest to równanie kwadratowe, to wyświetl odpowiedni komunikat. Przykładowe wartości współczynników równania kwadratowego oraz otrzymane pierwiastki przedstawia Tabela 6.

Tabela 6. Przykładowe współczynniki i pierwiastki równania kwadratowego

a	b	c	delta	x₁	x₂
2	-8	6	16	1	3
2	-4	2	0	1	
2	-2	1	-4	brak	

9. Przetwornik analogowo-cyfrowy zamienia sygnał analogowy (ciągły) na sygnał cyfrowy (dyskretny). Sygnał cyfrowy jest liczbą całkowitą, której zakres zależy od liczby bitów przetwornika. Na przykład przetwornik 10-bitowy zwraca wartości od 0 do 1023. Napisz program, w którym użytkownik wprowadza z klawiatury liczbę bitów, a program oblicza i wyświetla liczbę bajtów potrzebnych do zapisania tej liczby bitów. Przyjmij, że 1 bajt to 8 bitów. Zastosuj operator warunkowy.
10. Wczytaj z klawiatury wartości dwóch rezystancji **R₁** i **R₂** (liczby typu rzeczywistego). Wczytaj z klawiatury liczbę **x** (typ całkowity). Jeśli wartość **x** wynosi **0** to oblicz rezystancję zastępczą szeregowego połączenia rezystancji **R₁** i **R₂**. Jeśli wartość **x** wynosi **1** to oblicz rezystancję zastępczą równoległego połączenia rezystancji **R₁** i **R₂**. Zastosuj operator warunkowy. Wyświetl na ekranie wartość rezystancji zastępczej.
11. Wskaźniki zadziałania wkładek bezpiecznikowych oznacza się odpowiednimi kolorami zależnie od ich prądu znamionowego (Tabela 7).

Tabela 7. Wybrane kolory wskaźników zadziałania wkładek bezpiecznikowych

Barwa wskaźnika	Prąd znamionowy wkładki
zielona	6
czerwona	10
szara	16
niebieska	20

Napisz program, w którym po wprowadzeniu przez użytkownika prądu znamionowego wkładki, wyświetlana jest barwa odpowiadającego jej wskaźnika zadziałania. W przypadku błędnej wartości prądu wyświetl odpowiedni komunikat. Zastosuj instrukcję **switch**.

12. Taryfa G12, to taryfa dwustrefowa, która dzieli dobę na dwie strefy. Strefa dzienna (droższa) to godziny od 6:00 do 13:00 oraz od 15:00 do 22:00. Strefa nocna (tańsza) to godziny od 22:00 do 6:00 oraz od 13:00 do 15:00. Napisz program, w którym użytkownik wprowadza godzinę jako liczbę całkowitą (0 - 23). Program powinien wypisać odpowiedni tekst:

- „taryfa dzienna” - jeśli podana godzina należy do ze strefy „diennej”;
- „taryfa nocna” - jeśli podana godzina należy do strefy „nocnej”;
- „blad” - jeśli wprowadzona wartość nie mieści się w zakresie 0 - 23.

Zastosuj instrukcję **switch**. Godziny graniczne przypisz strefy nocnej.

13. Napisz program, w którym użytkownik wprowadza numer miesiąca, a program wyświetla nazwy wszystkich miesięcy, które pozostały od tego miesiąca do końca roku. Wyświetl odpowiedni komunikat w przypadku błędnego numeru miesiąca. Zastosuj instrukcję **switch**.

Przykładowe wywołanie programu:

```
Podaj numer miesiaca: 9
-----
wrzesien
pazdziernik
listopad
grudzien
```

4. Literatura

- [1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.
- [2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.
- [3] Deitel P.J., Deitel H.: Język C. Solidna wiedza w praktyce. Wydanie VIII. Helion, Gliwice, 2020.
- [4] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.
- [5] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.
- [6] Stańczyk J.: Nowoczesny C: przegląd C23 z przykładami. Helion, Gliwice, 2023.
- [7] <http://www.cplusplus.com/reference/clibrary> - C library - C++ Reference
- [8] <https://cpp0x.pl/dokumentacja/standard-C/1> - Standard C
- [9] <https://code.visualstudio.com/> - Visual Studio Code

5. Pytania kontrolne

1. Omów składnię i zastosowanie instrukcji warunkowej **if**.
2. Omów operatory relacyjne (porównania) i logiczne w języku C.
3. Opisz sposób tworzenia i obliczania wyrażeń logicznych.
4. Omów sposób wykonywania programu przy zagnieżdżaniu instrukcji **if-else**.
5. Omów zasadę działania operatora warunkowego **? :**. W jaki sposób operator warunkowy może zastępować instrukcję **if-else**?
6. Omów składnię i zasadę działania instrukcji wyboru wielowariantowego **switch**.
7. Wyjaśnij, w jakim celu w instrukcji **switch** stosowane są instrukcje **break**?

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.

- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.