



Politechnika Białostocka  
Wydział Elektryczny  
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja  
do pracowni specjalistycznej z przedmiotu

## **Programowanie C**

Kod przedmiotu: **CP1S01005**

(studia stacjonarne)

# **JĘZYK C - INSTRUKCJA WARUNKOWA IF, OPERATORY RELACYJNE I LOGICZNE, WYRAŻENIA LOGICZNE, ZAGNIEŹDŻANIE IF-ELSE**

Numer ćwiczenia

**PRC\_03**

Autor:  
dr inż. Jarosław Forenc

Białystok 2024

# Spis treści

<b>1. Opis stanowiska .....</b>	<b>3</b>
1.1. Stosowana aparatura .....	3
1.2. Oprogramowanie.....	3
<b>2. Wiadomości teoretyczne.....</b>	<b>3</b>
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
<b>3. Przebieg ćwiczenia.....</b>	<b>13</b>
<b>4. Literatura.....</b>	<b>15</b>
<b>5. Pytania kontrolne .....</b>	<b>16</b>
<b>6. Wymagania BHP.....</b>	<b>16</b>

---

**Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.**

© Wydział Elektryczny, Politechnika Białostocka, 2024 (wersja 1.1)

Wszelkie prawa zastrzeżone. Żadna część tej publikacji nie może być kopiowana i odtwarzana w jakiegokolwiek formie i przy użyciu jakichkolwiek środków bez zgody posiadacza praw autorskich.

# 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 1.92 (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
>	<b>a &gt; b</b>	<b>a</b> większe od <b>b</b>
<	<b>a &lt; b</b>	<b>a</b> mniejsze od <b>b</b>
>=	<b>a &gt;= b</b>	<b>a</b> większe lub równe <b>b</b>
<=	<b>a &lt;= b</b>	<b>a</b> mniejsze lub równe <b>b</b>
==	<b>a == b</b>	<b>a</b> równe <b>b</b>
!=	<b>a != b</b>	<b>a</b> nierówne <b>b</b> ( <b>a</b> różne od <b>b</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 <b>0</b> , a argument równy zero na wartość <b>1</b>
&&	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

<b>a</b>	<b>!a</b>
falsz	prawda
prawda	falsz

Tabela 4. Operatory koniunkcji i alternatywy

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>	<b>a    b</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)

<b>Operator</b>	<b>Typ operatora</b>
<b>!</b>	logiczny
<b>* / %</b>	arytmetyczne
<b>+ -</b>	arytmetyczne
<b>&gt; &lt; &gt;= &lt;=</b>	relacyjne
<b>== !=</b>	relacyjne
<b>&amp;&amp;</b>	logiczny
<b>  </b>	logiczny
<b>=</b>	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 &gt;= 0 &amp;&amp; j &lt;= 4)    (j &gt; 6 &amp;&amp; j &lt; 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.

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

## 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 zagnieżdżenia **if-else**:

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

### 3. Przebieg ćwiczenia

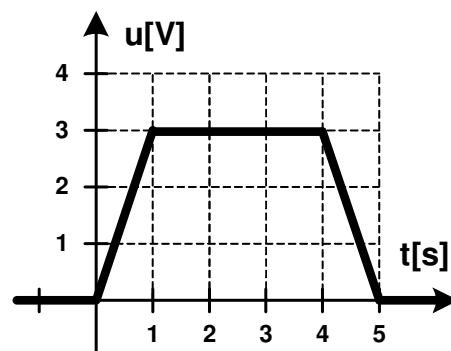
Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

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 wzrost w cm, a program wyświetla informację o zaliczeniu osoby do jednej z trzech grup:
  - poniżej 150 cm - wzrost niski;
  - od 150 cm, poniżej 180 cm - wzrost średni;
  - 180 cm i więcej - wzrost wysoki.
3. Napisz program wczytujący trzy liczby typu `int`. Wyświetl wartość największej oraz najmniejszej liczby.
4. Napisz program, w którym użytkownik wprowadza z klawiatury trzy liczby, a program wyświetla je od największej do najmniejszej, a następnie od najmniejszej do największej.
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 oraz dowolna liczba `x`. Jeśli dolna granica jest większa od

górną, 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. Napisz program wczytujący z klawiatury trzy liczby typu **int**, a następnie obliczający średnią arytmetyczną tylko tych liczb, które są **większe od zera**. Zabezpiecz program przed ewentualnym dzieleniem przez zero.
8. Na Rys. 1 przedstawiony jest przebieg 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

9. Napisz program rozwiązujący równanie kwadratowe:

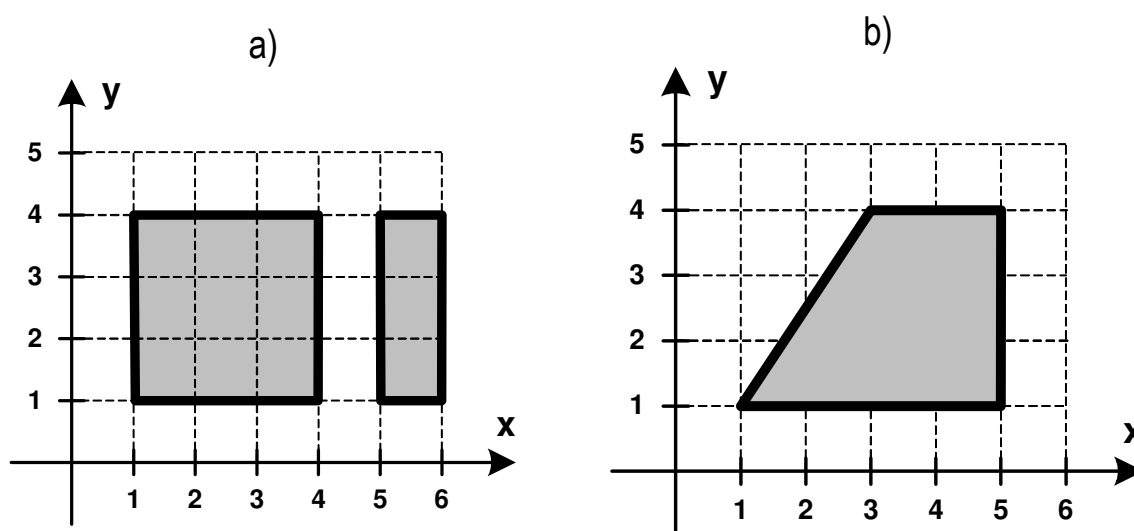
$$ax^2 + bx + c = 0 \quad (1)$$

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	$x_1$	$x_2$
2	-8	6	16	1	3
2	-4	2	0	1	
2	-2	1	-4	brak	

10. Napisz program sprawdzający, czy punkt o współrzędnych  $(x, y)$  wprowadzonych z klawiatury leży w obszarze zaznaczonym na Rys. 2 (do obszaru zaliczamy także jego granicę).



Rys. 2. Oznaczenie obszarów do zadania 10

## 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] <http://www.cplusplus.com/reference/clibrary> - C library - C++ Reference
- [7] <https://cpp0x.pl/dokumentacja/standard-C/1> - Standard C
- [8] <https://code.visualstudio.com/> - Visual Studio Code
- [9] <https://sourceforge.net/projects/mingw/> - MinGW

## 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**.

## 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.