Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Instrukcja
do pracowni specjalistycznej z przedmiotu

Informatyka 1

Kod przedmiotu: **EZ1D200 008**

(studia niestacjonarne)

JĘZYK C - INSTRUKCJA ITERACYJNA FOR, ZAGNIEŻDŻANIE PĘTLI FOR, INSTRUKCJE CONTINUE, BREAK I GOTO

Numer ćwiczenia

INF05Z

Autor:
dr inż. Jarosław Forenc

Białystok 2017

Spis treści

1. Opis stanowiska ... 3
1.1. Stosowana aparatura ... 3
1.2. Oprogramowanie ... 3
2. Wiadomości teoretyczne ... 3
2.1. Instrukcja for ... 3
2.2. Operatory zwiększania (++) i zmniejszania (--) ... 7
2.3. Dodatkowe uwagi do instrukcji for ... 8
2.4. Zagnieżdżanie pętli for ... 10
2.5. Instrukcja kontynuacji (continue) ... 12
2.6. Instrukcja break ... 14
2.7. Instrukcja skoku (goto) ... 14
3. Przebieg ćwiczenia ... 15
4. Literatura ... 17
5. Pytania kontrolne ... 17
6. Wymagania BHP ... 18

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows (XP/7/10).

1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Microsoft Visual Studio 2008 Standard Edition lub Microsoft Visual Studio 2008 Express Edition zawierające kompilator Microsoft Visual C++ 2008.

2. Wiadomości teoretyczne

2.1. Instrukcja for

W programach komputerowych bardzo często pewien fragment programu wykonywany jest wielokrotnie lub dla zmiennych przyjmujących kolejne wartości. W takich przypadkach stosowane są pętle czyli instrukcje iteracyjne. Podstawową instrukcją iteracyjną jest pętla **for**. W poniższym programie zastosowano pętlę **for** do pięciokrotnego wyświetlenia tego samego tekstu

Program wyświetlający pięć razy ten sam tekst.

```
#include <stdio.h>

int main(void)
{
    int i;

    for (i = 0; i < 5; i = i + 1)
        printf("Programowanie nie jest trudne\n");

    return 0;
}
```


Wynikiem działania programu będzie następujący wydruk:

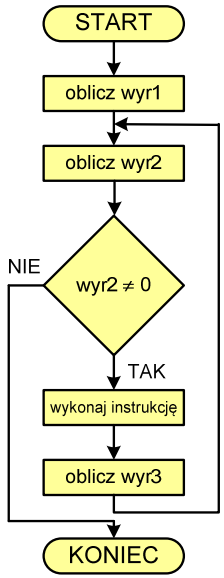
```
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
```

Ogólna postać instrukcji **for** jest następująca:

```
for (wyr1; wyr2; wyr3)
    instrukcja;
```

Instrukcja **for** tworzy pętlę pracującą w następujący sposób:

1. Wyliczane jest **wyr1** (jeśli występuje).
2. Wyliczane jest **wyr2**. Jeśli jego wartość jest różna od zera (czyli jest ono prawdziwe), to następuje przejście do kroku 3. W przeciwnym przypadku instrukcja **for** kończy działanie.
3. Wykonywana jest **instrukcja**.
4. Wyliczana jest wartość **wyr3**. Działanie jest wznawiane od kroku 2, czyli obliczania **wyr2**.



W pierwszym programie zmienna występująca w pętli **for** (**i**) pełniła tylko rolę tzw. licznika pętli. W kolejnym programie zmienna ta jest wykorzystywana także w obliczeniach.

Poniższy program zamienia temperaturę ze skali Fahrenheita na temperaturę w skali Celsjusza. W jednym programie zamieniane są wartości od 0 F do 50 F co 10 F.

Program przeliczający temperaturę ze skali Fahrenheita na skalę Celsjusza.

```
#include <stdio.h>
```

```c
int main(void)
{
    int f;

    for (f = 0; f < 60; f = f + 10)
        printf("%2d F  to  %6.2f C\n",f,5*(f-32.0)/9);

    return 0;
}
```

Wynikiem działania programu jest następujący wydruk:

```
 0 F  to  -17.78 C
10 F  to  -12.22 C
20 F  to   -6.67 C
30 F  to   -1.11 C
40 F  to    4.44 C
50 F  to   10.00 C
```

W nawiasach instrukcji **for** występują trzy części (wyrażenia) oddzielone od siebie średnikami:

- pierwsza część inicjalizuje pętlę (**f = 0**), jest ona wykonywana tylko raz, przed wejściem do pętli;
- druga część jest warunkiem sterującym powtarzaniem pętli (**f < 60**). Jeśli jest on prawdziwy to wykonywana jest instrukcja **printf()** znajdująca się w kolejnym wierszu kodu programu;
- następnie, po wykonaniu instrukcji **printf()**, wykonywana jest trzecia część, w której zwiększana jest wartość zmiennej wykorzystywanej w pętli (**f = f + 10**) i ponownie sprawdzany jest warunek sterujący powtarzaniem pętli (**f < 60**).

Zmienne występujące w pętlach nazywane są kolejnymi literami alfabetu: **i**, **j**, **k**, **l**, ..., chyba, że z kontekstu programu wynika użycie zmiennej o innej nazwie (tak jak w przypadku programu zamieniającego temperatury: **f** - zmienna przechowująca temperaturę w skali Fahrenheita). Zmienne te mogą przyjmować dowolne wartości, zależnie od trzech wyrażeń w nawiasach pętli **for**. Poniżej podano przykładowe pętle, w których instrukcja **printf()** wyświetla kolejne wartości przyjmowane przez zmienną całkowitą **i**.


```c
for (i = 0; i < 10; i = i + 1)
    printf("%d ",i);
```

```
0 1 2 3 4 5 6 7 8 9
```

```c
for (i = 1; i < 10; i = i + 2)
    printf("%d ",i);
```

```
1 3 5 7 9
```

```c
for (i = 10; i > 0; i = i - 1)
    printf("%d ",i);
```

```
10 9 8 7 6 5 4 3 2 1
```

```c
for (i = -9; i <= 9; i = i + 3)
    printf("%d ",i);
```

```
-9 -6 -3 0 3 6 9
```

Jeśli w każdej iteracji wartość zmiennej sterującej pętli jest zwiększana lub zmniejszana o **jeden** to zamiast zapisów:

```c
for (i = 0; i < 10; i = i + 1)
    instrukcja;

for (i = 10; i > 0; i = i - 1)
    instrukcja;
```

stosuje się operatory zwiększania (**++**) i zmniejszania (**--**), nazywane także operatorami inkrementacji i dekrementacji:

```
for (i = 0; i < 10; i++)
    instrukcja;

for (i = 10; i > 0; i--)
    instrukcja;
```

Operatory te zostały opisane w kolejnym rozdziale instrukcji.

2.2. Operatory zwiększania (++) i zmniejszania (--)

Operator **++** służy do zwiększania wartości zmiennej o **1**, zaś operator **--** służy do zmniejszania wartości zmiennej o **1**. Operatory te są jednoargumentowe. Można stosować je tylko do zmiennych (nie można ich stosować do wyrażeń). Operatory te mogą występować jako przedrostek lub przyrostek (Tabela 1).

Tabela 1. Operatory ++ i --

Zapis	Operator	Znaczenie
`++i`	preinkrementacji	Operator występuje przed nazwą zmiennej (i), wartość zmiennej modyfikowana jest przed jej użyciem.
`--i`	predekrementacji	
`i++`	postinkrementacji	Operator występuje po nazwie zmiennej (i), wartość zmiennej modyfikowana jest po użyciu jej poprzedniej wartości.
`i--`	postdekrementacji	

Rozpatrzmy następujący fragment programu:

```
int i = 2, j;

j = 2 * ++i;
printf("%d %d",i,j);
```

W powyższym programie najpierw wartość zmiennej **i** jest zwiększana o jeden (z **2** do **3**). Następnie wykonywana jest operacja mnożenia (**2 * 3**). Wynik tej operacji (**6**) przypisywany jest zmiennej **j**. Zatem instrukcja **printf()** wyświetli wartości: **3 6**.

Jeśli operator preinkrementacji zastąpimy operatorem postinkrementacji:

```
int i = 2, j;

j = 2 * i++;
printf("%d %d",i,j);
```

to najpierw wykonywana jest operacja mnożenia (**2 * i = 2 * 2**). Wynik tej operacji (**4**) przypisywany jest zmiennej **j**. Następnie zmienna **i** zwiększana jest o jeden (z **2** do **3**). Instrukcja **printf()** wyświetli zatem: **3 4**.

Miejsce umieszczenia operatorów inkrementacji i dekrementacji nie ma znaczenia w przypadku instrukcji typu:

```
i++;
++i;        równoważne:    i = i + 1;

i--;
--i;        równoważne:    i = i - 1;
```

gdyż efekt końcowy będzie taki sam (zwiększenie lub zmniejszenie wartości zmiennej **i** o 1).

Nie jest zalecane stosowanie operatorów **++** i **--** do zmiennej, która pojawia się w wyrażeniu więcej niż jeden raz. Wynik poniższej instrukcji:

```
i = i++;
```

jest według standardu języka C **niezdefiniowany**.

2.3. Dodatkowe uwagi do instrukcji for

Każde z trzech wyrażeń w nawiasach pętli **for** jest opcjonalne (może jego nie być), ale nawiasy i średniki są obowiązkowe.

```
for (wyr1; wyr2; wyr3)
    instrukcja;
```

Jeśli nie jest podane **wyr2**, to przyjmuje się, że jest ono prawdziwe. Zatem w prosty sposób można skonstruować pętlę nieskończoną.

```
for ( ; ; )
    instrukcja;
```

Jeśli w pętli **for** ma być wykonana więcej niż jedna instrukcja, to należy zastosować **instrukcję złożoną**, czyli objąć wszystkie te instrukcje nawiasami klamrowymi. W poniższym programie w pętli **for** wykonywane są trzy instrukcje.

Program obliczający średnią arytmetyczną 6 liczb całkowitych.

```
#include <stdio.h>
#pragma warning(disable:4996)

int main(void)
{
    int   i, x;
    float suma = 0.0f;

    for (i = 0; i < 6; i++)
    {
        printf("Podaj liczbe nr %d: ",i+1);
        scanf("%d",&x);
        suma = suma + x;
    }
    printf("Srednia: %.3f\n",suma/6);

    return 0;
}
```

Przykładowy wynik działania programu:

```
Podaj liczbe nr 1: 3
Podaj liczbe nr 2: 8
Podaj liczbe nr 3: 5
Podaj liczbe nr 4: 7
Podaj liczbe nr 5: 3
Podaj liczbe nr 6: 2
Srednia: 4.667
```


Po nawiasie zamykającym pętli **for** nie stawia się średnika. Konstrukcja ze średnikiem na końcu jest poprawna składniowo (kompilator nie zasygnalizuje błędu), ale oznacza wielokrotne wykonanie **instrukcji pustej** (w poniższym przykładzie - 10 razy). Natomiast **instrukcja** zostanie wykonana tylko jeden raz.

```
for (i = 0; i < 10; i++);
    instrukcja;
```

Często popełnianym błędem przez początkujących programistów jest wprowadzenie przecinków zamiast średników.

```
for (i = 0, i < 10, i++)
    instrukcja;
```

W takiej sytuacji kompilator wyświetli błąd:

```
1>c:\prog.cpp() : error C2143: syntax error : missing ';' before ')'
1>c:\prog.cpp() : error C2143: syntax error : missing ';' before ')'
```

Jeszcze innym rodzajem błędu jest podanie niewłaściwego warunku kontynuacji pętli. W poniższym przykładzie funkcja **printf()** nie wykona się ani razu.

```
for (i = 0; i > 10; i++)
    printf("%d ",i);
```

W następnym przykładzie otrzymujemy pętlę nieskończoną, gdyż warunek **i > 0** jest zawsze prawdziwy:

```
for (i = 1; i > 0; i++)
    printf("%d ",i);
```

2.4. Zagnieżdżanie pętli for

Zagnieżdżanie pętli **for** polega na tym, że jako instrukcja w pętli występuje kolejna pętla **for**. Pierwsza pętla nazywana jest pętlą **zewnętrzną**, zaś druga -

wewnętrzną. Zasada działania zagnieżdżonych pętli zostanie pokazana na przykładzie poniższego kodu programu.

```
for (i = 1; i <= 3; i++)
    for (j = 1; j <= 2; j++)
        printf("i = %d, j = %d\n",i,j);
```

Wynikiem działania zagnieżdżonych pętli jest wyświetlenie tekstu:

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
i = 3, j = 1
i = 3, j = 2
```

W zewnętrznej pętli zmienna **i** otrzymuje wartość **1**. Następnie wykonywana jest pętla wewnętrzna, w której zmienna **j** przyjmuje wartości **1** i **2**. Po zakończeniu pętli wewnętrznej następuje powrót do pętli zewnętrznej - zmienna **i** jest zwiększana o **1**, przyjmując wartość **2**. Ponownie wykonywana jest pętla wewnętrzna, itd.

W kolejnym programie zagnieżdżanie pętli zostało wykorzystane do wyświetlenia na ekranie tabliczki mnożenia.

Program wyświetlający tabliczkę mnożenia.

```
#include <stdio.h>

int main(void)
{
    int i, j;

    for (i = 1; i < 11; i++)
    {
        for (j=1; j<11; j++)
            printf("%2d  ",i*j);
        printf("\n");
    }

    return 0;
}
```


Wynik działania powyższego programu:

```
 1   2   3   4   5   6   7   8   9  10
 2   4   6   8  10  12  14  16  18  20
 3   6   9  12  15  18  21  24  27  30
 4   8  12  16  20  24  28  32  36  40
 5  10  15  20  25  30  35  40  45  50
 6  12  18  24  30  36  42  48  54  60
 7  14  21  28  35  42  49  56  63  70
 8  16  24  32  40  48  56  64  72  80
 9  18  27  36  45  54  63  72  81  90
 0  20  30  40  50  60  70  80  90 100
```

Zewnętrzna pętla **for** wykonywana jest 10 razy (**i** zmienia się od 1 do 10). Wewnętrzna pętla **for** wykonywana jest także 10 razy (**j** zmienia się od 1 do 10), zatem instrukcja **printf()** będzie wykonana 100 razy.

2.5. Instrukcja kontynuacji (continue)

Instrukcja kontynuacji używana jest wewnątrz pętli **for** i ma postać:

```
continue;
```

Instrukcja ta powoduje bezwarunkowe przejście na początek pętli (do następnej iteracji) z pominięciem instrukcji znajdujących się poniżej **continue**. Dokładniej mówiąc, powoduje przejście do wyliczania wartości **wyr3** w pętli **for**.

Program obliczający średnią arytmetyczną 10 liczb całkowitych wprowadzonych z klawiatury przy uwzględnieniu tylko liczb nieujemnych.

```
#include <stdio.h>
#pragma warning(disable:4996)

int main(void)
{
    int   i, x, ilosc = 0;
    float suma = 0.0f;

    for (i = 0; i < 10; i++)
    {
        printf("Podaj liczbe nr %d: ",i+1);
        scanf("%d",&x);
```

```
        if (x < 0)
            continue;
        suma = suma + x;
        ilosc++;
    }

    if (ilosc > 0)
    {
        printf("Ilosc liczb:   %d\n",ilosc);
        printf("Suma liczb:    %g\n",suma);
        printf("Srednia:       %.3f\n",suma/ilosc);
    }
    else
        printf("Wszystkie liczby sa ujemne\n");

    return 0;
}
```

Przykładowy wynik działania programu:

```
Podaj liczbe nr 1: 3
Podaj liczbe nr 2: -4
Podaj liczbe nr 3: 2
Podaj liczbe nr 4: -4
Podaj liczbe nr 5: 0
Podaj liczbe nr 6: 2
Podaj liczbe nr 7: 1
Podaj liczbe nr 8: -3
Podaj liczbe nr 9: 9
Podaj liczbe nr 10: 2
Ilosc liczb:   7
Suma liczb:    19
Srednia:       2.714
```

W powyższym programie jeśli wartość kolejnej liczby **x** jest mniejsza od zera to wywoływana jest instrukcja **continue**. Powoduje ona pominięcie dwóch instrukcji z bieżącej iteracji:

```
suma = suma + x;
ilosc++;
```

i przejście do kolejnej iteracji (wprowadzania kolejnej liczby).

2.6. Instrukcja break

Instrukcja **break** umożliwia wcześniejsze zakończenie pętli **for**. Wywołanie tej instrukcji powoduje natychmiastowe opuszczenie pętli i przejście do instrukcji znajdującej się bezpośrednio po pętli **for**.

```
for (i = 1; i < 10; i++)
{
    if (i % 5 == 0)
        break;
    printf("%d\n",i);
}
printf("Koniec, i = %d\n",i);
```

Gdy warunek w instrukcji **if** będzie prawdziwy zostanie wykonana instrukcja **break** powodująca przerwanie pętli **for**:

```
1
2
3
4
Koniec, i = 5
```

W przypadku zagnieżdżonych pętli przerywane jest działanie tylko jednej pętli - najbardziej wewnętrznej.

2.7. Instrukcja skoku (goto)

Instrukcja ta w postaci:

```
goto label;
```

przekazuje sterowanie do miejsca w programie oznaczonego etykietą o nazwie **label**. Etykietą można oznaczać tylko całe instrukcje. Skok może odbywać się tylko w ramach tej samej funkcji.

Wystąpienie w poniższym fragmencie programu instrukcji skoku powoduje przeniesienie sterowania do instrukcji rozpoczynającej się od etykiety **dalej**.

```
for (i = 1; i < 10; i++)
{
    if (i % 5 == 0)
        goto dalej;
    printf("%d\n",i);
}

dalej: printf("Koniec, i = %d\n",i);
```

Wynik działania powyższego fragmentu programu:

```
1
2
3
4
Koniec, i = 5
```

W programach w języku C **nie zaleca się** stosowania instrukcji **goto**. Jedynym przypadkiem, kiedy dopuszcza się zastosowanie **goto**, jest wyjście z układu wielu zagnieżdżonych pętli.

3. Przebieg ćwiczenia

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

1. Napisz program wyświetlający na ekranie poniższe liczby. Do wyświetlenia jednego wiersza liczb zastosuj jedną pętlę **for**.

```
1   2   3   4   5   6   7   8   9
9   8   7   6   5   4   3   2   1
2   4   6   8   10   12   14   16
-12   -8   -4   0   4   8   12
1   2   4   8   16   32   64   128
11   9.5   8   6.5   5   3.5   2   0.5
```

2. Napisz program obliczający i wyświetlający wartość silni liczby **n** wprowadzonej z klawiatury.

3. Rezystancję przewodu w zależności od temperatury opisuje wzór (1).

$$R_t = R_{20}\left[1 + \alpha\left(t - 20°C\right)\right] \tag{1}$$

gdzie:

R_t - rezystancja w temperaturze t,

R_{20} - rezystancja w temperaturze 20 ºC,

α - współczynnik temperaturowy rezystancji w $ºC^{-1}$.

Przewód miedziany ($\alpha = 4{,}3 \cdot 10^{-3}$) ma w temperaturze t = 20 ºC rezystancję $R_{20} = 10\ \Omega$. Napisz program obliczający i wyświetlający rezystancję tego przewodu w temperaturze od 0 ºC do 200 ºC z krokiem 20 ºC.

4. Napisz program, w którym użytkownik wprowadza dwie liczby całkowite określające dolną i górną granicę przedziału. Program powinien wyświetlić wszystkie liczby z tego przedziału, ich kwadraty i sześciany. Przykład:

```
Dolna granica:  2
Gorna granica:  5

2   4   8
3   9   27
4   16  64
5   25  125
```

5. Ciąg Fibonacciego opisany jest następującym wzorem rekurencyjnym:

$$F_n = \begin{cases} 0 & \text{dla } n = 0 \\ 1 & \text{dla } n = 1 \\ F_{n-1} + F_{n-2} & \text{dla } n > 1 \end{cases} \tag{2}$$

Napisz program obliczający wartość **n**-tego wyrazu tego ciągu.

6. Napisz program, który prosi o podanie liczby całkowitej z zakresu ⟨**1, 15**⟩. Jeśli liczba znajduje się w zakresie, to program wyświetla na ekranie trójkąt ze znaków **@**. Na przykład, dla podanej liczby **4** na ekranie powinno pojawić się:

```
@
@@
@@@
@@@@
```

7. Obwód elektryczny ma następujące parametry:

$$R = 15\ \Omega,\ L = 0{,}125\ H,\ C = 0{,}254\ mF$$

Wyznacz pulsację rezonansową (rezonans prądów) obwodu jeśli jego admitancja zastępcza określona jest poniższym wzorem:

$$\underline{Y} = \frac{R}{R^2 + X_C^2} + \mathrm{j}\left(\frac{X_C}{R^2 + X_C^2} - \frac{1}{X_L}\right) \tag{3}$$

Wskazówka: zbadaj pulsację w zakresie od 100 do 500 rad/s z krokiem 0,1 rad/s.

4. Literatura

[1] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[2] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[3] Prinz P., Crawford T.: Język C w pigułce. APN Promise, Warszawa, 2016.

[4] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.

[5] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.

[6] Wileczek R.: Microsoft Visual C++ 2008. Tworzenie aplikacji dla Windows. Helion, Gliwice, 2009.

5. Pytania kontrolne

1. Scharakteryzuj operatory inkrementacji i dekrementacji.
2. Omów składnię i zastosowanie pętli **for**.
3. Wyjaśnij, jaką rolę w pętli **for** mogą pełnić instrukcje **break**, **goto**, **continue**.
4. Na dowolnym przykładzie opisz sposób wykonywania zagnieżdżonych pętli **for**.

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.