



Politechnika Białostocka
Wdział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

Programowanie w języku C

Kod przedmiotu: **TZ1E1004BK**

(studia niestacjonarne)

JĘZYK C - INSTRUKCJA SWITCH, INSTRUKCJA ITERACYJNA FOR, ZAGNIEŹDŻANIE PĘTLI FOR

Numer ćwiczenia

PROGwC_04

Autor:
dr inż. Jarosław Forenc

Białystok 2024

Spis treści

1. Opis stanowiska	3
1.1. Stosowana aparatura	3
1.2. Oprogramowanie	3
2. Wiadomości teoretyczne.....	3
2.1. Instrukcja wyboru wielowariantowego - switch.....	3
2.2. Instrukcja for.....	8
2.3. Operatory zwiększania (++) i zmniejszania (--).....	12
2.4. Dodatkowe uwagi do instrukcji for	13
2.5. Zagnieżdżanie pętli for	15
2.6. Instrukcja kontynuacji (continue)	17
2.7. Instrukcja break.....	18
3. Przebieg ćwiczenia.....	19
4. Literatura.....	21
5. Pytania kontrolne	22
6. Wymagania BHP	22

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

© Wydział Elektryczny, Politechnika Białostocka, 2024 (wersja 1.0)

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 zainstalowane jest środowisko programistyczne Code::Blocks.

2. Wiadomości teoretyczne

2.1. 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_stale: instrukcje;
    case wyrażenie_stale: instrukcje;
    case wyrażenie_stale: 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 podstawia jego kod pod zmienną **key**. Następnie w instrukcji **switch** kod klawisza porównywany jest z wyrażeniami stałymi znajdującymi się po **case**. Jeśli wciśniętym klawiszem był **'+'**, to zmienne **x1** i **x2** dodawane są do siebie i wyświetlana jest ich suma. Jeśli wciśnięto **'-'**, 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: 2
Ocena negatywna
```

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

2.2. 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:

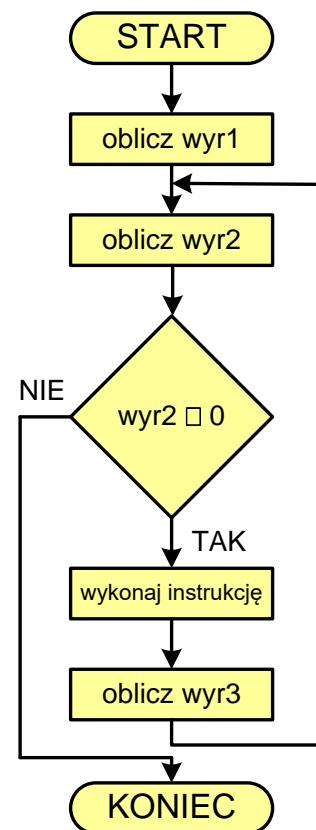
```
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
```

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>
```

```
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**.

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

0 1 2 3 4 5 6 7 8 9

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

1 3 5 7 9

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

10 9 8 7 6 5 4 3 2 1

```
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:

```
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.

2.3. 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**.

2.4. Dodatkowe uwagi do instrukcji **for**

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>

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 (w tym przypadku Code::Blocks) wyświetli następujące ostrzeżenie i błędy:

```
c:\prog\main.c | | warning: right-hand operand of comma expression  
                has no effect [-Wunused-value]
```

```
c:\prog\main.c | | error: expected ';' before ')' token
```

```
c:\prog\main.c | | error: expected expression before ')' token
```

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);
```

2.5. 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.

2.6. 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>

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).

2.7. 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.

3. Przebieg ćwiczenia

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

1. Wskaźniki zadziałania wkładek bezpiecznikowych oznacza się odpowiednimi kolorami zależnie od ich prądu znamionowego (Tabela 2).

Tabela 2. 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**.

2. Napisz program, w którym użytkownik wprowadza numer dnia tygodnia. Program powinien wypisać tekst:

- „zwykły dzień” - dla dni od poniedziałku do piątku;
- „weekend” - dla soboty i niedzieli;
- „bledny numer dnia” - dla pozostałych wartości.

Przyjmij: 1 - poniedziałek, 2 - wtorek, 3 - środa, itd. Zastosuj instrukcję **switch**.

3. Napisz program, w którym użytkownik wprowadza rok oraz numer miesiąca, a program wyświetla liczbę dni w podanym miesiącu. Uwzględnij obecność lat przestępnych. Zastosuj instrukcję **switch**. Uwaga: rok przestępny (liczbowo) spełnia następujące warunki: jest podzielny przez 4, ale nie jest podzielny przez 100 lub jest podzielny przez 400.

4. 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
```

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

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

$$R_t = R_{20} [1 + \alpha(t - 20^\circ\text{C})] \quad (1)$$

gdzie:

R_t - rezystancja w temperaturze t ,

R_{20} - rezystancja w temperaturze 20°C ,

α - współczynnik temperaturowy rezystancji w $^\circ\text{C}^{-1}$.

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

7. 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 sześciiany. Przykład:

Dolna granica: 2

Gorna granica: 5

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

8. Ciąg Fibonacciego opisany jest następującym wzorem rekurencyjnym:

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

Napisz program obliczający wartość n-tego wyrazu tego ciągu.

9. Napisz program, który prosi o podanie liczby całkowitej z zakresu $\langle 1, 15 \rangle$. 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:

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

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://www.codeblocks.org/> - Code::Blocks

5. Pytania kontrolne

1. Omów składnię i zasadę działania instrukcji wyboru wielowariantowego **switch**.
2. Wyjaśnij, w jakim celu w instrukcji **switch** stosowane są instrukcje **break**?
3. Scharakteryzuj operatory inkrementacji i dekrementacji.
4. Omów składnię i zastosowanie pętli **for**.
5. Wyjaśnij, jaką rolę w pętli **for** mogą pełnić instrukcje **break** i **continue**.
6. 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.