Język C – zajęcia nr 5

Instrukcja wyboru `switch`

`switch (` ***wyr*** **`)`** ***inst***

Głównym przeznaczeniem instrukcji **`switch`** jest rozgałęzianie wykonania programu na wiele różnych ścieżek w zależności od wartości pewnego wyrażenia. Pełne wykorzystanie możliwości instrukcji **`switch`** następuje, gdy instrukcja ***inst*** jest instrukcją złożoną. Każda z instrukcji składających się na instrukcję złożoną może posiadać etykietę mającą formę:

`case` ***wyr_stałe*** **`:`**

gdzie ***wyr_stałe*** jest stałym wyrażeniem, które dla różnych etykiet (przypadków case) ma różne wartości. Jedna z instrukcji w instrukcji złożonej może mieć etykietę:

`default :`

Wykonanie instrukcji **`switch`** polega na wyznaczeniu wartości wyrażenia ***wyr***, a następnie na porównywaniu tej wartości z wartościami ***wyr_stałe*** kolejnych etykiet **`case`**. Jeżeli wartość stałego wyrażenia pewnej etykiety jest równa wartości wyrażenia ***wyr***, to od instrukcji związanej z tą etykietą rozpoczyna się sekwencyjne wykonywanie instrukcji składających się na instrukcję złożoną. Jeżeli wartości stałych wyrażeń wszystkich etykiet są różne od wartości wyrażenia ***wyr***, to sekwencyjne wykonywanie instrukcji składających się na instrukcję złożoną rozpoczyna się od instrukcji z etykietą **`default`**, a gdy brak jest takiej etykiety, to żadna instrukcja zawarta w instrukcji **`switch`** nie jest wykonywana. Sekwencyjne wykonywanie instrukcji zawartych w instrukcji **`switch`** zostaje zakończone po osiągnięciu końca bloku lub wykonaniu instrukcji **`break`**.

Wprowadź i uruchom program, zinterpretuj kod źródłowy:

```
#include <stdio.h>
int main()
{
    int n;
    printf("\nPodaj liczbe calkowita z zakresu 1..9: ");
    scanf("%d",&n);
    switch (n)
    {
        case 1: printf(" jeden");
                break;
        case 2: printf(" dwa");
                break;
        case 3: printf(" trzy");
                break;
        case 4: printf(" cztery");
                break;
        case 5: printf(" piec");
                break;
        case 6: printf(" szesc");
                break;
        case 7: printf(" siedem");
                break;
        case 8: printf(" osiem");
                break;
        case 9: printf(" dziewiec");
                break;
       default: printf(" Nie rozpoznana liczba");
    }
    getch(); return 0;
}
```

Uwaga: Instrukcja w bloku instrukcji **`switch`** może być poprzedzona wieloma etykietami, co pozwala na tworzenie złożonych warunków rozgałęzienia wykonania programu.

```
#include <stdio.h>
int main()
{
    char c;
    printf("\nPodaj jeden znak ASCII: ");
    scanf("%c",&c);
    printf("Wczytany znak to ");
    switch (c)
    {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'y': printf("mala samogloska");
                  break;
        case 'A':
        case 'E':
        case 'I':
        case 'O':
        case 'U':
        case 'Y': printf("wielka samogloska");
                  break;
        case 'b': case 'c': case 'd':
```

```
        case 'f': case 'g': case 'h':
        case 'j': case 'k': case 'l':
        case 'm': case 'n': case 'p':
        case 'q': case 'r': case 's':
        case 't': case 'v': case 'w':
        case 'x': case 'z': printf("mala spolgloska");
                  break;
        case 'B': case 'C': case 'D':
        case 'F': case 'G': case 'H':
        case 'J': case 'K': case 'L':
        case 'M': case 'N': case 'P':
        case 'Q': case 'R': case 'S':
        case 'T': case 'V': case 'W':
        case 'X': case 'Z': printf("wielka spolgloska");
                  break;
        case '0': case '1':
        case '2': case '3':
        case '4': case '5':
        case '6': case '7':
        case '8': case '9': printf("cyfra");
                  break;
         default: printf("nie litera i nie cyfra");
    }
    getch(); return 0;
}
```

Instrukcja `break`

```
break ;
```

Instrukcja może występować w instrukcji złożonej należącej do instrukcji **`switch`**, **`for`**, **`while`** lub **`do-while`**. Wykonanie instrukcji **`break`** powoduje natychmiastowe przerwanie wykonywania instrukcji **`switch`**, **`for`**, **`while`** lub **`do-while`**, w której ta instrukcja **`break`** występuje.

Uwaga: Jeżeli instrukcja złożona, w której występuje **`break`**, jest zagnieżdżona w innej instrukcji **`switch`**, **`for`**, **`while`** lub **`do-while`**, to przerwanie wykonania dotyczy najbardziej wewnętrznej instrukcji **`switch`**, **`for`**, **`while`** lub **`do-while`**.

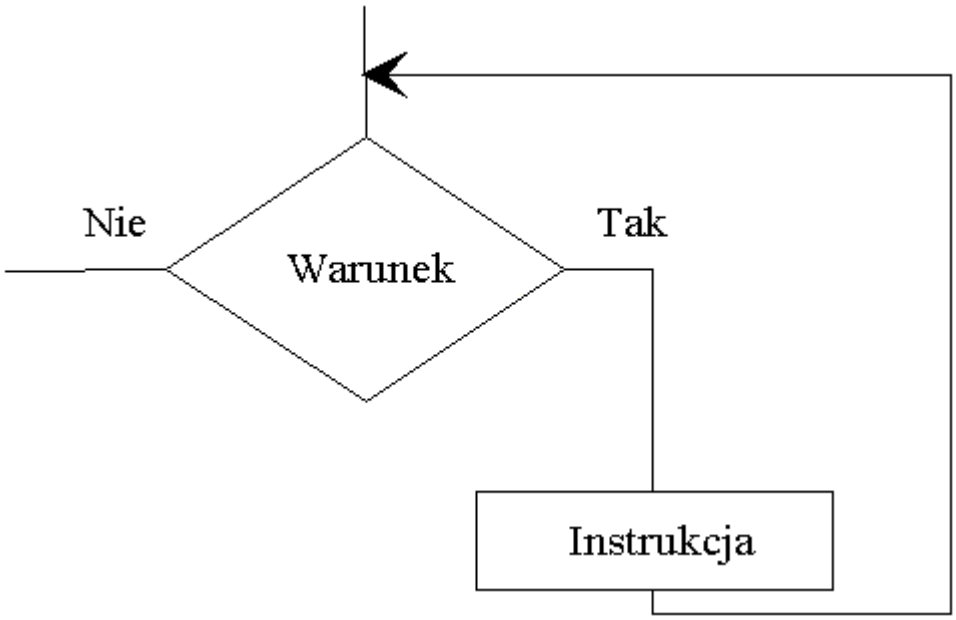
Algorytmy iteracyjne

Iteracja (pętla) - konstrukcja umożliwiająca wykonanie pewnej części algorytmu więcej niż jeden raz. Algorytm, w którym występuje pętla określa się mianem **algorytmu iteracyjnego**.

Dwa podstawowe rodzaje pętli:

a) pętla z licznikiem,
b) pętla warunkowa
- ze sprawdzaniem wartości warunku na początku pętli,
- ze sprawdzaniem wartości warunku na końcu pętli.

Pętla warunkowa ze sprawdzaniem warunku na początku pętli



Instrukcja iteracji `while`

`while (`*wyr*`)` ***inst***

Instrukcja umożliwia organizowanie pętli poprzez powtarzanie wykonania instrukcji ***inst***. Wykonywane są czynności:

1. Wyliczana jest wartość wyrażenia ***wyr***.
2. Jeżeli jego wartość jest **różna od zera**, to wykonywana jest instrukcja ***inst***. W przeciwnym przypadku wykonywanie instrukcji **`while`** zostaje zakończone.
3. Czynności powtarzane są od kroku 1.

Uwaga: Jeżeli instrukcja ***inst*** jest instrukcją złożoną, to występująca wewnątrz instrukcja **`break`** lub **`return`** może zakończyć wykonywanie instrukcji **`while`**.

Algorytm Euklidesa:

Dla uzyskania największego wspólnego dzielnika **nwd** dwóch niezerowych liczb naturalnych **m** oraz **n**, należy postępować następująco:

1. Jeżeli **m** jest równe **n**, to jako **nwd** przyjąć wartość **n** i zatrzymać wykonanie algorytmu, w przeciwnym przypadku wykonać krok **2**.

2. Jeżeli wartość **m** jest większa od **n**, to zastąpić **m** przez różnicę **m-n** i wykonać krok **1**, w przeciwnym przypadku zastąpić **n** przez różnicę **n-m** i wykonać krok **1**.

C05-1. Wprowadź i uruchom program realizujący algorytm Euklidesa:

```
#include <stdio.h>
int main()
{
    int k1,k2,p;
    printf("Podaj dwie liczby naturalne: ");
    scanf("%d%d",&k1,&k2);
    while(k1!=k2)
    {
        p= k1>k2 ? k1-k2 : k2-k1;
        if(k1>k2) k1=p;
            else  k2=p;
    }
    printf("Najwiekszym wspolnym podzielnikiem jest %d",k1);
    getch(); return 0;
}
```

```
     KLAWIATURA   432 810↵
Podaj dwie liczby naturalne: 432 810
Najwiekszym wspolnym podzielnikiem jest 54
```

Zadania – napisz i uruchom programy:

1. **C05-2.** Wczytaj z klawiatury znak reprezentujący jeden z dwuargumentowych operatorów arytmetycznych **oper** (+ – * / %), a następnie dwie liczby całkowite **a** i **b**. Wypisz wynik operacji **a oper b**. W przypadku gdy znak nie reprezentuje operatora arytmetycznego, zgłoś komunikat „Bledny operator”. Sprawdź także, czy w przypadku operatorów / i % nie zachodzi dzielenie przez 0, jeśli tak – wypisz odpowiedni komunikat. Wskazówka: zastosuj instrukcje **switch** i **if**.
2. **C05-3.** Zmodyfikuj program C05-2 (tworząc osobny program) tak, aby działał w pętli obliczając wyniki operacji dla kolejnych par liczb dopóty, dopóki użytkownik zamiast znaku operatora **oper** wprowadzi znak **q**. Wskazówka: zastosuj instrukcję **while**. Do wczytania **oper** użyj getch(), por. materiały C02.

Kody źródłowe programów C05-1, C05-2, C05-3, wraz z oknami zawierającymi przykładowe efekty uruchomienia tych programów, należy zapisać w **jednym** dokumencie (format Word) o nazwie **C05.doc** i przesłać za pośrednictwem platformy Moodle w wyznaczonym terminie.