



Podstawy Programowania

dr inż. Tomasz Marciniak

Wykład 2

- Instrukcje

Operacje wejścia/wyjścia

- **printf()** - wyświetla dane na ekranie
- **puts()** - wyświetla łańcuch na ekranie
- **scanf()** - pobiera dane z klawiatury
- **getche()** - pobiera znak z klawiatury i daje echo
- **gets()** - pobiera łańcuch z klawiatury

Przykład

```
//przykład 1  
#include <stdio.h>  
int main()  
{  
    printf("Drukujemy na ekranie");  
    return 0;  
}
```



komentarz

Przykład

```
//przykład 1
#include <stdio.h>
int main()
{
    printf("Drukujemy na ekranie");
    return 0;
}
```

Dyrektywa preprocesora
Dołącza plik nagłówkowy

Przykład

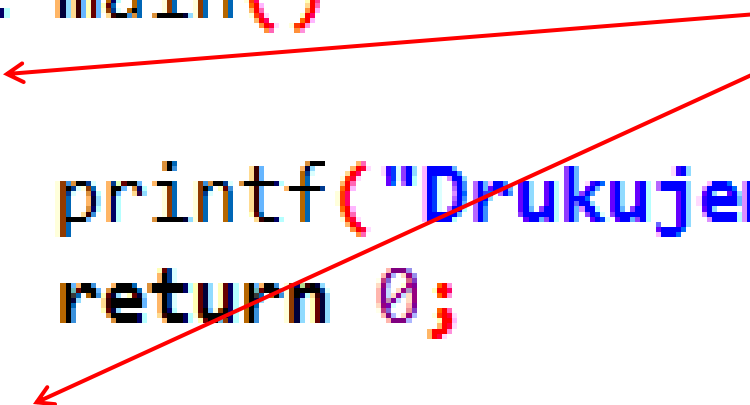
```
//przykład 1
#include <stdio.h>
int main()
{
    printf("Drukujemy na ekranie");
    return 0;
}
```

Funkcja main()
Program główny

Przykład

```
//przykład 1
#include <stdio.h>
int main()
{
    printf("Drukujemy na ekranie");
    return 0;
}
```

Nawiasy ograniczające
blok funkcji



Przykład

```
//przykład 1
#include <stdio.h>
int main()
{
    printf("Drukujemy na ekranie");
    return 0;
}
```

Blok funkcji



```
printf("Drukujemy na ekranie");
return 0;
```


Działa!! Ale ekran znika

//przykład 1

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

```
#include <conio.h>
```

```
int main()
```

```
{
```

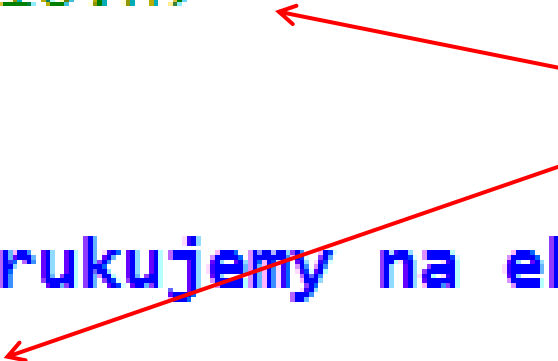
```
    printf("Drukujemy na ekranie");
```

```
    getch();
```

```
    return 0;
```

```
}
```

Czytanie znaku z
klawiatury



Działa!! Ale ekran znika

//przykład 1

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

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    printf("Drukujemy na ekranie");
```

```
    getch();
```

```
    return 0;
```

```
}
```

C:\Users\Tomek\Documents\Wyk | ady\Progra

Drukujemy na ekranie

A gdyby wydrukować kilka linii?

//przykład 3

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

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    printf("Drukujemy na ekranie");
```

```
    printf("Drukujemy na ekranie");
```

```
    printf("Drukujemy na ekranie");
```

```
    getch();
```

```
    return 0;
```

```
}
```

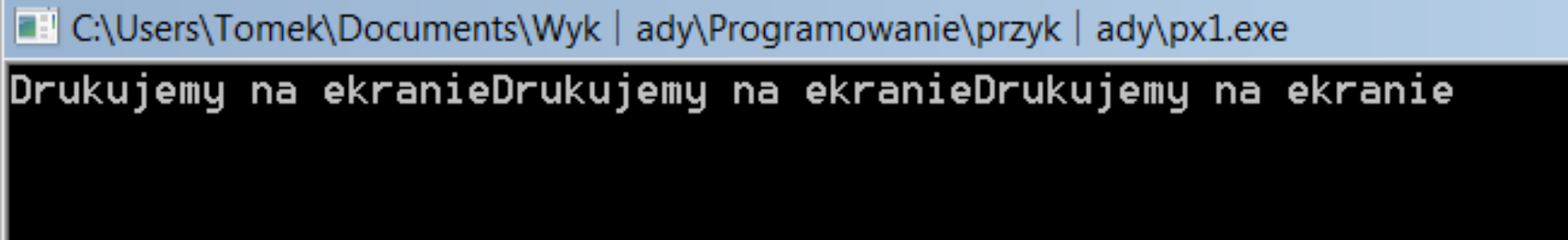
A gdyby wydrukować kilka linii?

//przykład 3

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

```
#include <conio.h>
```

```
int main()
```

A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\Tomek\Documents\Wyk | ady\Programowanie\przyk | ady\px1.exe". The command prompt displays the text "Drukujemy na ekranie" three times, indicating the output of the program.

C:\Users\Tomek\Documents\Wyk | ady\Programowanie\przyk | ady\px1.exe

Drukujemy na ekranieDrukujemy na ekranieDrukujemy na ekranie

```
printf("Drukujemy na ekranie");
```

```
getche();
```

```
return 0;
```

```
}
```

Kody „ukośnika”

- `\a` dzwonek
- `\n` nowa linia
- `\r` powrót karetki
- `\t` tabulator poziomy
- `\v` tabulator pionowy

A gdyby wydrukować kilka linii?

//przykład 3

```
#include <stdio.h>
#include <conio.h>
int main()
{
    printf("Drukujemy na ekranie\n");
    printf("Drukujemy na ekranie\n");
    printf("Drukujemy na ekranie\n");
    getch();
    return 0;
}
```

A gdyby wydrukować kilka linii?

//przykład 3

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

```
# C:\Users\Tomek\Documents\Wyk | ady\Pro
```

```
i Drukujemy na ekranie
```

```
{ Drukujemy na ekranie
```

```
  Drukujemy na ekranie
```

```
");
```

```
");
```

```
");
```

```
return 0;
```

```
}
```

Inny przykład

//przykład 5

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

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    char nazwa[20];
```

```
    int cena;
```

```
    printf("\npodaj nazwe :")
```

```
    gets (nazwa);
```

```
    printf("podaj cene :");
```

```
    scanf("%d",&cena);
```

```
    printf("\nCena dla %s to: %d", nazwa,cena);
```

```
    getch();
```

```
    return 0;
```

```
}
```

Czyta znaki z klawiatury
(w C nie ma łańcuchów)

Inny przykład

//przykład 5

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

```
#include <conio.h>
```

```
int main()
```

```
{
```

```
    char nazwa[20];
```

```
    int cena;
```

```
    printf("\npodaj nazwe :");
```

```
    gets (nazwa);
```

```
    printf("podaj cene :");
```

```
    scanf("%d",&cena);
```

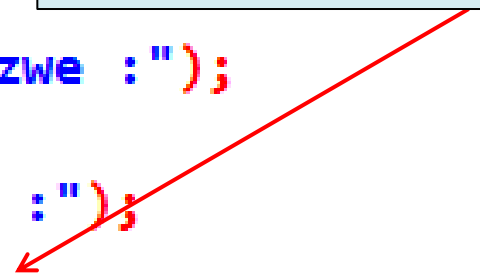
```
    printf("\nCena dla %s to: %d", nazwa,cena);
```

```
    getch();
```

```
    return 0;
```

```
}
```

Czyta znaki z klawiatury
i konwertuje zgodnie z typem



Inny przykład

//przykład 5

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

```
#include <conio.h>
```

```
int main  
{
```

```
    ch
```

```
    in
```

```
    pr
```

```
    ge
```

```
    pr
```

```
    sc
```

```
    pr
```

```
    ge
```

```
    return 0;
```

```
}
```



C:\Users\Tomek\Documents\W

podaj nazwe :abc

podaj cene :123

Cena dla abc to: 123

a,cena);

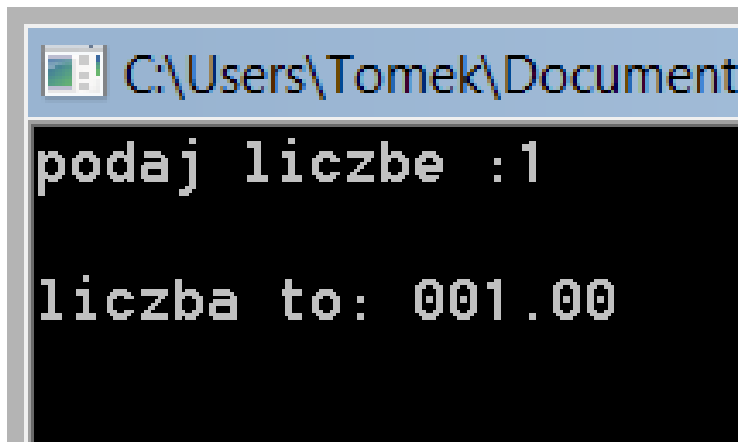
printf() (p6.c)

- %s znaki
- %d liczby całkowite
- %f float
- %x hex

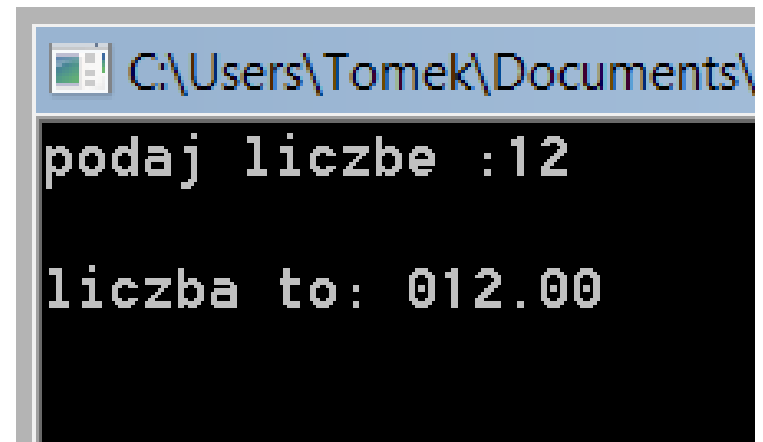
printf() - przykład

```
#include <stdio.h>
#include <conio.h>
int main()
{
    float cena;
    printf("podaj liczbe :");
    scanf("%f",&cena);
    printf("\nliczba to: %06.2f",cena);
    getch();
    return 0;
}
```

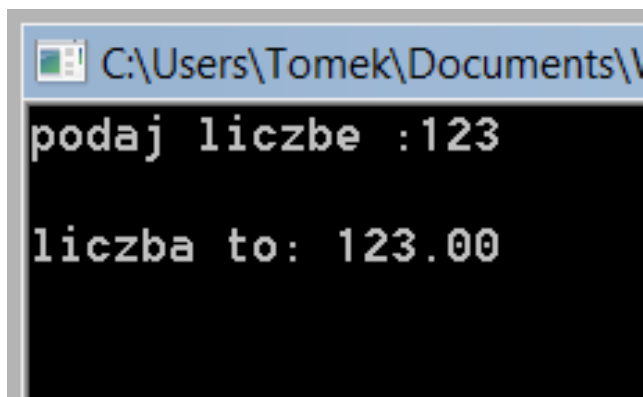
printf() - przykład



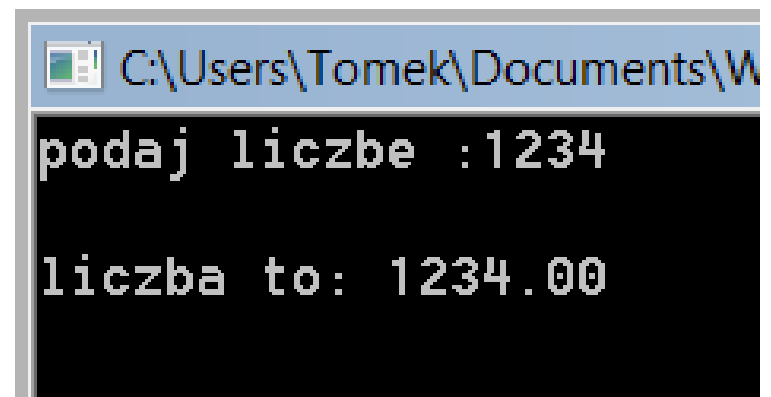
```
podaj liczbe :1  
liczba to: 001.00
```



```
podaj liczbe :12  
liczba to: 012.00
```



```
podaj liczbe :123  
liczba to: 123.00
```



```
podaj liczbe :1234  
liczba to: 1234.00
```

Instrukcja warunkowa

- Najprostsza forma instrukcji `if` jest następująca:

```
if (wyrażenie)  
    instrukcja;
```

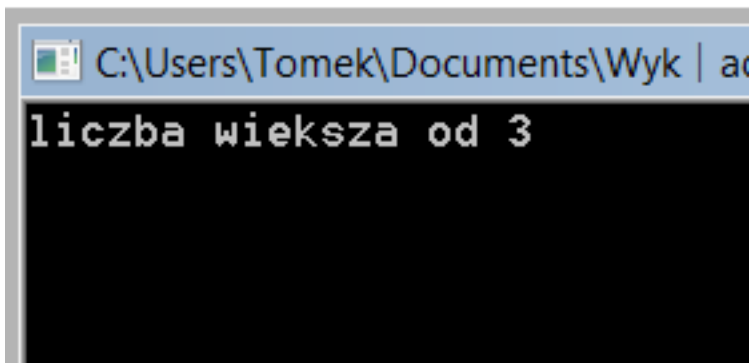
Ważne: Nie należy stawiać średnika za nawiasem wyrażenia!!

Wyrażenie w nawiasach może być całkowicie dowolne, ale najczęściej jest to jedno z wyrażeń **relacji**. Jeśli wyrażenie to ma wartość **false**, wtedy **instrukcja** jest pomijana. Jeśli wyrażenie jest prawdziwe (ma wartość **true**), wtedy **instrukcja** jest wykonywana.

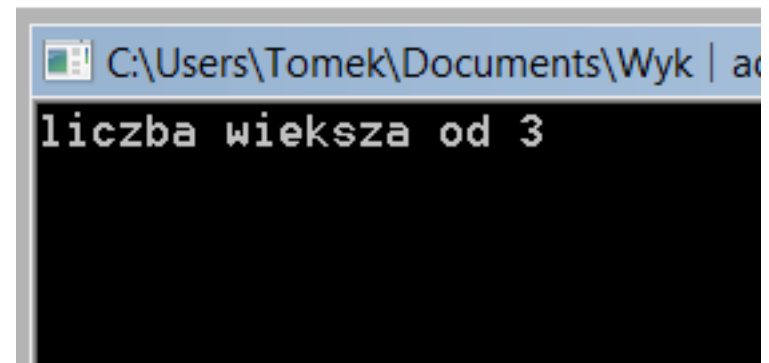
Instrukcja warunkowa

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a=4;
    if (a>3)
        printf("liczba wieksza od 3");
    getch();
    return 0;
}
```

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a=2;
    if (a>3);
        printf("liczba wieksza od 3");
    getch();
    return 0;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Users\Tomek\Documents\Wyk | ac". The command prompt displays the output "liczba wieksza od 3" in white text on a black background.

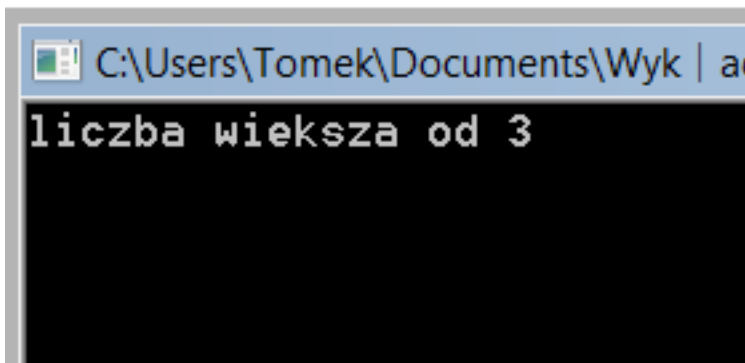


A screenshot of a Windows command prompt window. The title bar shows the path "C:\Users\Tomek\Documents\Wyk | ac". The command prompt displays the output "liczba wieksza od 3" in white text on a black background.

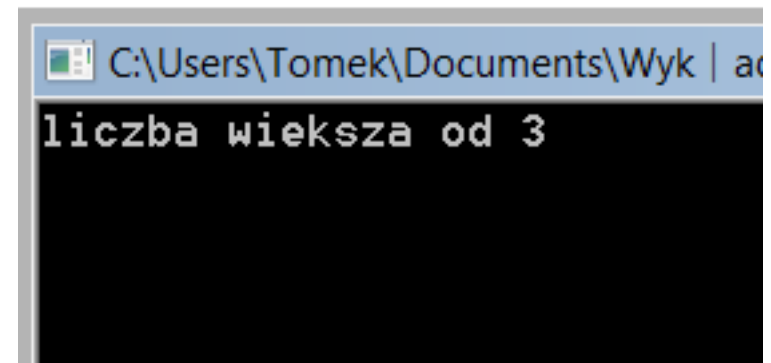
Instrukcja warunkowa

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a=4;
    if (a>3)
        printf("liczba wieksza od 3");
    getch();
    return 0;
}
```

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a=2;
    if (a>3); ←
        printf("liczba wieksza od 3");
    getch();
    return 0;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Users\Tomek\Documents\Wyk | ac". The command prompt displays the text "liczba wieksza od 3" on a single line.



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Users\Tomek\Documents\Wyk | ac". The command prompt displays the text "liczba wieksza od 3" on a single line.

if ... else

- Często zdarza się, że w swoim programie chcesz wykonać jakiś fragment kodu, jeżeli spełniony zostanie pewien warunek, oraz inny fragment kodu, gdy warunek ten nie zostanie spełniony.

- użycie słowa kluczowego `else`

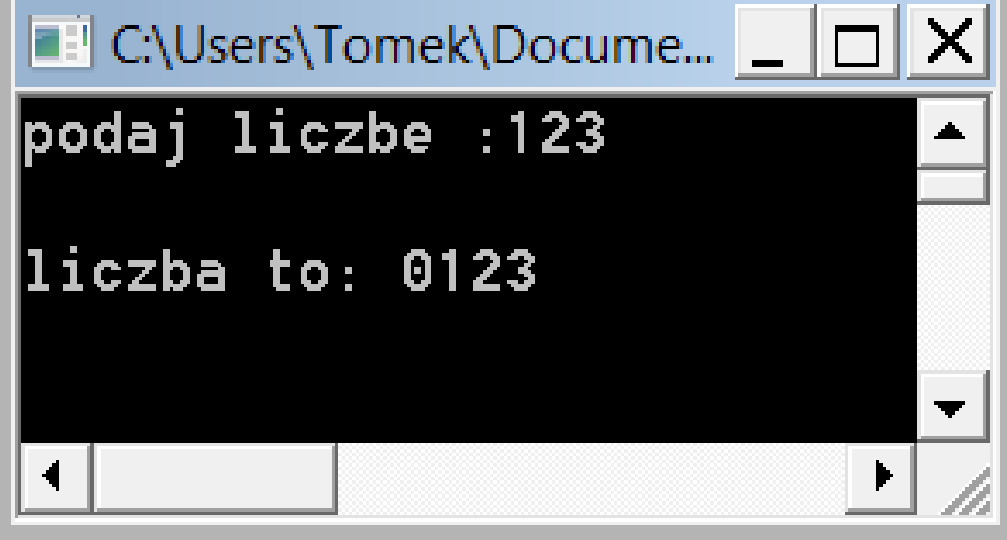
```
if (wyrażenie)
    instrukcja;
else
    instrukcja;
```

If,else (p7.c)

```
1 //Przykład 7
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     if (liczba>10)
10         printf("\nliczba to: %04d",liczba);
11     else
12         printf("Liczba za mala");
13     getch();
14     return 0;
15 }
```

If,else (p7.c)

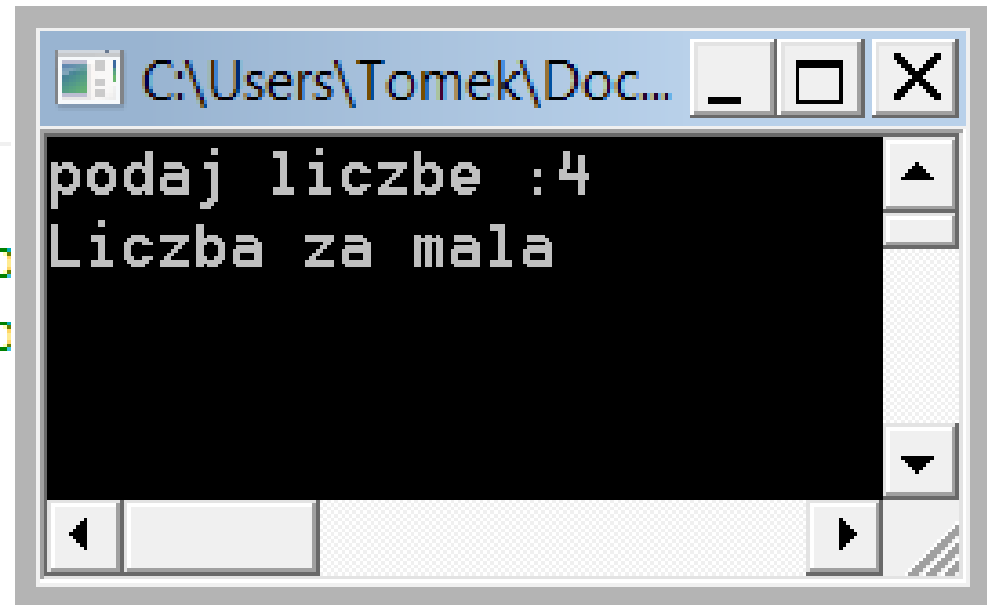
```
1  //Przykład 7
2  #include <stdio.h>
3  #include <conio.h>
4  int main()
5  {
6      int liczba;
7      printf("podaj liczbe :");
8      scanf("%d",&liczba);
9      if (liczba>10)
10         printf("\nliczba to: %04d",liczba);
11     else
12         printf("Liczba za mala");
13     getch();
14     return 0;
15 }
```



C:\Users\Tomek\Docume...
podaj liczbe :123
liczba to: 0123

If,else (p7.c)

```
1 //Przykład 7
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     if (liczba>10)
10         printf("\nliczba to: %04d",liczba);
11     else
12         printf("Liczba za mala");
13     getch();
14     return 0;
15 }
```



```
C:\Users\Tomek\Doc...
podaj liczbe :4
Liczba za mala
```

if – blok instrukcji {}

- Ponieważ blok instrukcji ujętych w nawiasy klamrowe stanowi odpowiednik instrukcji pojedynczej, warunkowo wykonywany fragment kodu może być dość rozbudowany:

```
if (wyrażenie)
{
    instrukcja1;
    instrukcja2;
    instrukcja3;
}
```

Zaawansowane instrukcje if

- Warto zauważyć, że w klauzuli `if` lub `else` może być zastosowana dowolna instrukcja, nawet inna instrukcja `if` lub `else`. Z tego powodu możemy natrafić na złożone instrukcje `if`, przyjmujące postać:

```
if (wyrażenie1)
{
    if (wyrażenie2)
        instrukcja1;
    else
    {
        if (wyrażenie3)
            instrukcja2;
        else
            instrukcja3;
    }
}
else
    instrukcja4;
```

Zaawansowane instrukcje if p8.c)

```
1 //Przykład 8
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     if (liczba>=10)
10    {
11        printf("\nliczba wieksza/rowna od 10");
12        if (liczba>=20)
13            printf("\nliczba wieksza/rowna od 20");
14        else
15        {
16            printf("\nLiczba mniejsza od 20");
17        }
18    }
19    else
20        printf("\nLiczba mniejsza od 10");
21    getch();
22    return 0;
23 }
```

Zaawansowane instrukcje if p8.c)

```
1 //Przykład 8
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     if (liczba>=10)
10    {
11        printf("\nliczba wieksza/rowna od 10");
12        if (liczba>=20)
13            printf("\nliczba wieksza/rowna od 20");
14        else
15        {
16            printf("\nLiczba mniejsza od 20");
17        }
18    }
19    else
20        printf("\nLiczba mniejsza od 10");
21    getch();
22    return 0;
23 }
```

Co pojawi się na ekranie po wprowadzeniu liczb 9,15,22?

Zaawansowane instrukcje if p8.c)

```
1 //Przykład 8
2 #include <stdio.h>
3 #include <conio.h>
4
5 podaj liczbe :9
6
7 Liczba mniejsza od 10
8
9
10
11 printf("\nliczba wieksza/rowna od 10 ");
12 if (liczba >= 20)
13     printf("\nliczba wieksza/rowna od 20");
14 else
15 {
16     printf
17 }
18 }
19 else
20     printf("\n
21 getch();
22 return 0;
23 }
```

C:\Users\Tomek\Documents\Wyk

podaj liczbe :9

Liczba mniejsza od 10

C:\Users\Tomek\Documents\Wyk | ady\

podaj liczbe :15

liczba wieksza/rowna od 10
Liczba mniejsza od 20

C:\Users\Tomek\Documents\Wyk | ady\Pro

podaj liczbe :22

liczba wieksza/rowna od 10
liczba wieksza/rowna od 20

Operatory logiczne

- Często zdarza się, że chcemy zadać więcej niż jedno relacyjne pytanie na raz. „Czy jest prawdą że x jest większe od y i czy jest jednocześnie prawdą, że y jest większe od z?” Aby móc podjąć działanie, program musi mieć możliwość sprawdzenia, czy oba te warunki są prawdziwe — lub czy przynajmniej któryś z nich jest prawdziwy.

Operator	Symbol	Przykład
I (AND)	&&	wyrażenie1 && wyrażenie2
LUB (OR)		wyrażenie1 wyrażenie2
NIE (NOT)	!	!wyrażenie

AND

Instrukcja logicznego I (AND) oblicza dwa wyrażenia, jeżeli oba mają wartość `true`, wartością całego wyrażenia I także jest `true`. Jeśli prawdą jest, że jesteś głodny I prawdą jest, że masz pieniądze, WTEDY możesz kupić obiad. Zatem

```
if ( (x == 5) && (y == 5) )
```

będzie prawdziwe, gdy zarówno `x`, jak i `y` ma wartość `5`, zaś będzie nieprawdziwe, gdy któraś z tych zmiennych będzie miała wartość różną od `5`. **Zapamiętaj, że aby całe wyrażenie było prawdziwe, prawdziwe muszą być oba wyrażenia.**

AND

Instrukcja logicznego I (AND) oblicza dwa wyrażenia, jeżeli oba mają wartość `true`, wartością całego wyrażenia I także jest `true`. Jeśli prawdą jest, że jesteś głodny I prawdą jest, że masz pieniądze, WTEDY możesz kupić obiad. Zatem

Na co musimy uważać ???!

```
if ( (x == 5) && (y == 5) )
```

będzie prawdziwe, gdy zarówno `x`, jak i `y` ma wartość `5`, zaś będzie nieprawdziwe, gdy któraś z tych zmiennych będzie miała wartość różną od `5`. **Zapamiętaj, że aby całe wyrażenie było prawdziwe, prawdziwe muszą być oba wyrażenia.**

OR

Instrukcja logicznego LUB (OR) oblicza dwa wyrażenia, gdy któreś z nich ma wartość `true`, wtedy wartością całego wyrażenia LUB także jest `true`. Jeśli prawdą jest, że masz gotówkę LUB prawdą jest że, masz kartę kredytową, WTEDY możesz zapłacić rachunek. Nie potrzebujesz jednocześnie gotówki i karty kredytowej, choć posiadanie obu jednocześnie nie przeszkadza. Zatem

```
if ( (x == 5) || (y == 5) )
```

będzie prawdziwe, gdy `x` lub `y` ma wartość `5`, lub gdy obie zmienne mają wartość `5`.

OR

Instrukcja logicznego LUB (OR) oblicza dwa wyrażenia, gdy któreś z nich ma wartość `true`, wtedy wartością całego wyrażenia LUB także jest `true`. Jeśli prawdą jest, że masz gotówkę LUB prawdą jest że, masz kartę kredytową, WTEDY możesz zapłacić rachunek. Nie potrzebujesz jednocześnie gotówki i karty kredytowej, choć posiadanie obu jednocześnie nie przeszkadza. Zatem

```
if ( (x == 5) || (y == 5) )
```

Na co musimy uważać ???!

będzie prawdziwe, gdy `x` lub `y` ma wartość `5`, lub gdy obie zmienne mają wartość `5`.

NOT

Instrukcja logicznego NIE (NOT) ma wartość `true`, gdy sprawdzane wyrażenie ma wartość `false`. Jeżeli sprawdzane wyrażenie ma wartość `true`, operator logiczny NIE zwraca wartość `false`. Zatem

```
if ( !(x == 5) )
```

jest prawdziwe tylko wtedy, gdy `x` jest różne od `5`.
Identycznie działa zapis:

```
if (x != 5)
```


Modyfikacja poprzedniego przykładu

Napisz program sprawdzający czy wprowadzona liczba całkowita należy do przedziału 0-9, 10-19, 20-29, 30-39, 40- nieskończoności.

Modyfikacja poprzedniego przykładu (p9)

```
1 //Przykład 9
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     if (liczba<10) printf("\nliczba mniejsza od 10");
10    if ((liczba>=10) && (liczba<20))printf("\nliczba wieksza/rowna 10 i mniejsza od 20");
11    if ((liczba>=20) && (liczba<30))printf("\nliczba wieksza/rowna 20 i mniejsza od 30");
12    if ((liczba>=30) && (liczba<40))printf("\nliczba wieksza/rowna 30 i mniejsza od 40");
13    if (liczba>=40) printf("\nliczba wieksza/rowna 40");
14    getch();
15    return 0;
16 }
```

Ćwiczenia na bitach

```
1 //Przykład px10.c
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     int i;
8
9     //tutaj działania na bitach
10    liczba=5;
11
12    //koniec
13
14    for (i = 31; i >= 0; i--){
15        if ((i + 1) % 8 == 0) printf(" ");
16        printf( "%1d", ((liczba >> i) % 2) );
17    }
18    getch();
19    return 0;
20 }
```

Instrukcja wyboru

- Składnia instrukcji **switch**

```
switch(wyrażenie)
{
case stała1 : /* ciąg_instrukcji1 */;break;
case stała2 : /* ciąg_instrukcji2 */;break;
default : /* ciąg_instrukcji0 */;break;
case stałax : /* ciąg_instrukcjiX */;;
}
```

gdzie wyrażenie jest dowolnym wyrażeniem całkowitym. Niektóre implementacje dopuszczają także wyrażenia innych typów, których wartość może być poddana automatycznej konwersji do typu całkowitego.

Instrukcja switch c.d.

- Należy pamiętać, że w przypadku braku instrukcji `break` na końcu bloku instrukcji (po `case`), wykonanie przechodzi także do następnego przypadku `case`. Czasem takie działanie jest zamierzone, ale zwykle jest po prostu błędem. Jeśli zdecydujesz się na wykonanie instrukcji w kilku kolejnych przypadkach `case`, pamiętaj o umieszczeniu obok komentarza, który wyjaśni, że nie pominąłeś instrukcji `break` przypadkowo.

Instrukcja switch c.d.

- Instrukcja `switch` umożliwia rozgałęzienie programu (w zależności od wartości wyrażenia). Na początku wykonywania instrukcji następuje obliczenie wartości `wyrażenia`, gdy odpowiada ona którejś z wartości przypadku `case`, wykonanie programu przechodzi do tego właśnie przypadku. Wykonywanie instrukcji jest kontynuowane aż do końca ciała instrukcji `switch` lub do czasu napotkania instrukcji `break`.

Instrukcja wyboru (p10.c)

```
1 //Przykład 10
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     switch (liczba)
10    {
11        case 0:printf("\nliczba 0");
12        case 1:printf("\nliczba 1");
13        case 2:printf("\nliczba 2");
14        default:printf("\nliczba > 2");
15    }
16
17     getch();
18     return 0;
19 }
```

Instrukcja wyboru (p10.c)

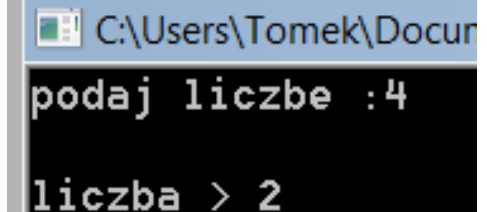
```
1 //Przykład 10
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     switch (liczba)
10    {
11        case 0:printf("\nliczba 0");
12        case 1:printf("\nliczba 1");
13        case 2:printf("\nliczba 2");
14        default:printf("\nliczba > 2");
15    }
16
17     getch();
18     return 0;
19 }
```

Czy wszystko zadziała tak jak chcemy ????

Instrukcja wyboru (p10.c)

```
1 //Przykład 10
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     switch (liczba)
10    {
11        case 0:printf("\nliczba 0");
12        case 1:printf("\nliczba 1");
13        case 2:printf("\nliczba 2");
14        default:printf("\nliczba > 2");
15    }
16
17     getch();
18     return 0;
19 }
```

Czy wszystko zadziała tak jak chcemy ????

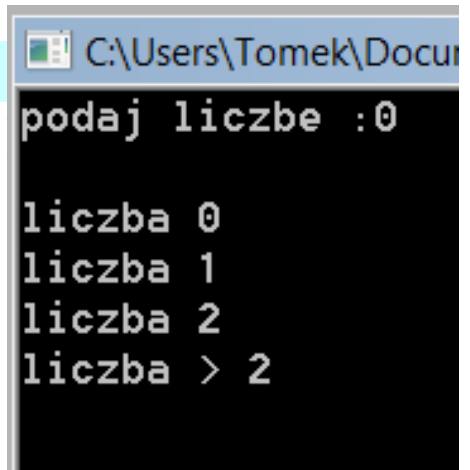


C:\Users\Tomek\Docur
podaj liczbe :4
liczba > 2

Instrukcja wyboru (p10.c)

```
1 //Przykład 10
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     switch (liczba)
10    {
11        case 0:printf("\nliczba 0");
12        case 1:printf("\nliczba 1");
13        case 2:printf("\nliczba 2");
14        default:printf("\nliczba > 2")
15    }
16
17     getch();
18     return 0;
19 }
```

Czy wszystko zadziała tak jak chcemy ????

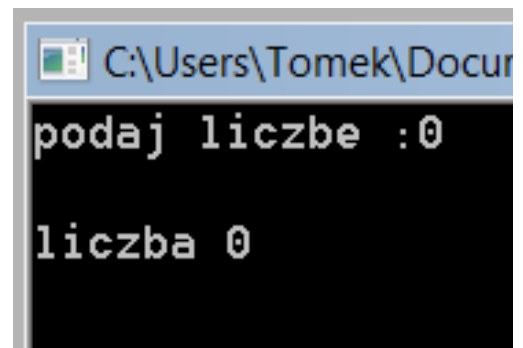


```
C:\Users\Tomek\Docum
podaj liczbe :0
liczba 0
liczba 1
liczba 2
liczba > 2
```

Instrukcja wyboru (p10.c)

```
1 //Przykład 10
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int liczba;
7     printf("podaj liczbe :");
8     scanf("%d",&liczba);
9     switch (liczba)
10    {
11        case 0:printf("\nliczba 0");break;
12        case 1:printf("\nliczba 1");break;
13        case 2:printf("\nliczba 2");break;
14        default:printf("\nliczba > 2");
15    }
16
17    getch();
18    return 0;
19 }
```

NIE ZAPOMNIJ O break !!!



```
C:\Users\Tomek\Docur
podaj liczbe :0
liczba 0
```

Pętle w programie

- pętla **for**
- pętla **while**
- pętla **do...while**

Instrukcja pętli

- Składnia instrukcji

```
for (początek pętli; koniec pętli; zmiana  
warunku)  
{  
    wyrażenie1;  
    wyrażenie2;  
}
```

Wykorzystywana jest do powtarzania wyrażeń znajdujących się w pętli określoną ilość razy.

Pętle for

- Składnia instrukcji **for** jest następująca:

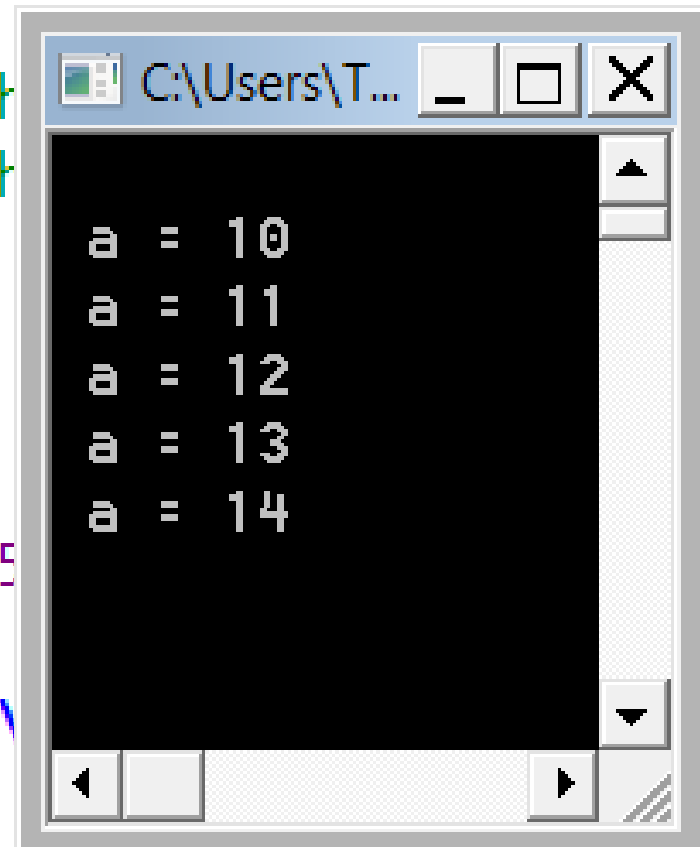
```
for (inicjalizacja; test; akcja )  
    instrukcja;
```
- Instrukcja **inicjalizacja** jest używana w celu zainicjalizowania stanu licznika lub innego przygotowania do wykonania pętli. Instrukcja **test** jest dowolnym wyrażeniem języka C/C++, które jest obliczane przed każdym wykonaniem zawartości pętli. Jeśli wyrażenie **test** ma wartość **true**, wykonywane jest ciało pętli, po czym wykonywana jest instrukcja **akcja** z nagłówka pętli (zwykle po prostu następuje inkrementacja zmiennej licznikowej).

for przykład (p11.c)- jaki wynik?

```
1 //Przykład 11
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int a = 10;
7     int i;
8     for (i=0; i<5; i++)
9     {
10         printf("\n a = %d",a++);
11     }
12     getch();
13     return 0;
14 }
```

for przykład (p11.c)- jaki wynik?

```
1 //Przykład 11
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int a = 10;
7     int i;
8     for (i=0; i<5
9     {
10         printf("\n
11     }
12     getch();
13     return 0;
14 }
```



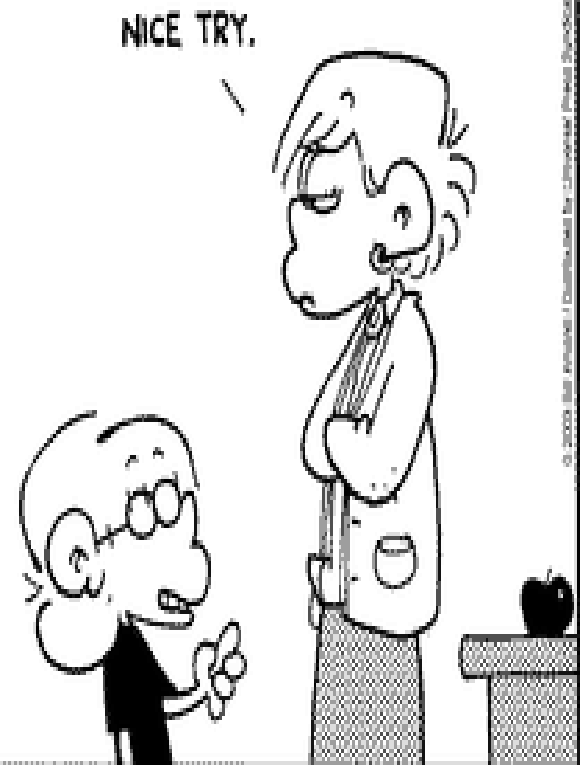
Jeszcze jeden przykład 😊

```
#include <stdio.h>
int main(void)
{
    int count;

    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");

    return 0;
}
```

WZROD 10-3



© 2000 All rights reserved. Distributed by Universal Uclick, Inc. All rights reserved.

Puste instrukcje w pętli for

- Każdą z instrukcji w nagłówku pętli `for` można pominąć. W tym celu należy oznaczyć jej położenie średnikiem (`;`).
- Ponieważ w samym nagłówku pętli `for` można wykonać tak wiele pracy, więc czasem ciało pętli może już niczego nie robić. Dlatego pamiętaj o zastosowaniu instrukcji pustej (`;`) jako ciała funkcji. Średnik może zostać umieszczony w tej samej linii, co nagłówek pętli, ale wtedy łatwo go przeoczyć.

Instrukcje puste w pętli for (p12.c)

- Ponieważ ciało pętli nie wykonuje żadnych czynności, użyto w nim instrukcji pustej (*;*).

```
for (int i = 0; i<5; printf("\n i = %d",i++))  
    ;
```

Pętle zagnieżdżone (p13.c)

- Pętle mogą być zagnieżdżone, tj. pętla może znajdować się w ciele innej pętli.
- Pętla wewnętrzna jest wykonywana wielokrotnie, przy każdym wykonaniu pętli zewnętrznej.

```
for (int i = 0; i<rows; i++)  
{  
    for (int j = 0; j<columns; j++)  
        printf("\n %d,%d)", i, j);  
}
```

Pętla while

- Pętla `while` (dopóki) powoduje powtarzanie zawartej w niej sekwencji instrukcji tak długo, jak długo zaczynające pętlę wyrażenie warunkowe pozostaje prawdziwe.

```
while(counter < 5) // sprawdzenie, czy warunek jest
                    spełniony
{
    counter++;
    .               // ciało pętli
    Instrukcje;
    .
}
```

Instrukcja while

- Składnia instrukcji `while` jest następująca:

```
while ( warunek )  
    instrukcja;
```

- `warunek` jest wyrażeniem języka C/C++, zaś `instrukcja` jest dowolną instrukcją lub blokiem instrukcji C/C++.

Opis instrukcji while (p14.c)

- Gdy wartością wyrażenia `warunek` jest `true` (prawda), wykonywana jest `instrukcja`, po czym następuje powrót do początku pętli i ponowne sprawdzenie warunku. Czynność ta powtarza się, dopóki `warunek` zwraca wartość `true`. Gdy wyrażenie `warunek` ma wartość `false`, działanie pętli `while` kończy się i program przechodzi do instrukcji następujących po pętli.
- Przykład:

```
// zliczanie do 10
int x = 0;
while (x < 10)
    printf("x=%d", x++);
```

Warunek while

- Warunek sprawdzany w pętli `while` może być złożony, tak jak każde poprawne wyrażenie języka C++. Może zawierać wyrażenia tworzone za pomocą operatorów logicznych `&&` (I), `||` (LUB) oraz `!` (NIE).

continue oraz break (p14.c)

- Może się zdarzyć, że przed wykonaniem całego zestawu instrukcji w pętli zaistnieje konieczność powrócenia do jej początku. Służy do tego instrukcja `continue` (kontynuuj).
- Może zdarzyć się także, że zaistnieje konieczność wyjścia z pętli jeszcze przed spełnieniem warunku końca. Instrukcja `break` (przerwij) powoduje natychmiastowe wyjście z pętli i przejście wykonywania do następnych instrukcji programu.

(Zmodyfikować przykład 14)

Uwaga!!

- Instrukcje `continue` oraz `break` powinny być używane ostrożnie. Wraz z `goto` stanowią one dwie najbardziej niebezpieczne instrukcje języka (są one niebezpieczne z tych samych powodów co instrukcja `goto`). Programy zmieniające nagle kierunek działania są trudniejsze do zrozumienia, a używanie instrukcji `continue` i `break` według własnego uznania może uniemożliwić analizę nawet niewielkich pętli `while`.

Pętla do...while

- Pętla `do...while` (wykonuj...dopóki) wykonuje ciało pętli przed sprawdzeniem warunku i sprawia że instrukcje w pętli zostaną wykonane co najmniej raz.
- Istnieje możliwość, że ciało pętli `while` nigdy nie zostanie wykonane. Instrukcja `while` sprawdza swój warunek przed wykonaniem którejkolwiek z zawartych w niej instrukcji, a gdy ten warunek nie jest spełniony, całe ciało pętli jest pomijane.

Do...while – przykład (p15.c)

```
1 //Przykład 15
2 #include <stdio.h>
3 #include <conio.h>
4 int main()
5 {
6     int i=0;
7     do{
8         printf("\n i = %d",i++);
9     }while (i<5);
10    getch();
11    return 0;
12 }
```

A teraz ... napisać program

- Mamy 3 sędziów : s1,s2,s3 , oceniających zawodników. Zawodnik przechodzi do następnej rundy jeśli przynajmniej 2 z sędziów zagłosowało na „tak”.