



TERMINY ZAJĘĆ: 9, 16, 23, 30 MARCA

16, 13 KWIETNIA 2019 ROKU,

GODZ. 9.00 – 12.15

MIEJSCE: WYDZIAŁ MATEMATYKI I

NAUK INFORMACYJNYCH

SALA: 311

PROWADZĄCY: BARTŁOMIEJ JOP

KONTAKT:

BARTEKJ_97@HOTMAIL.COM

C++ POZIOM PODSTAWOWY Z ELEMENTAMI ARDUINO

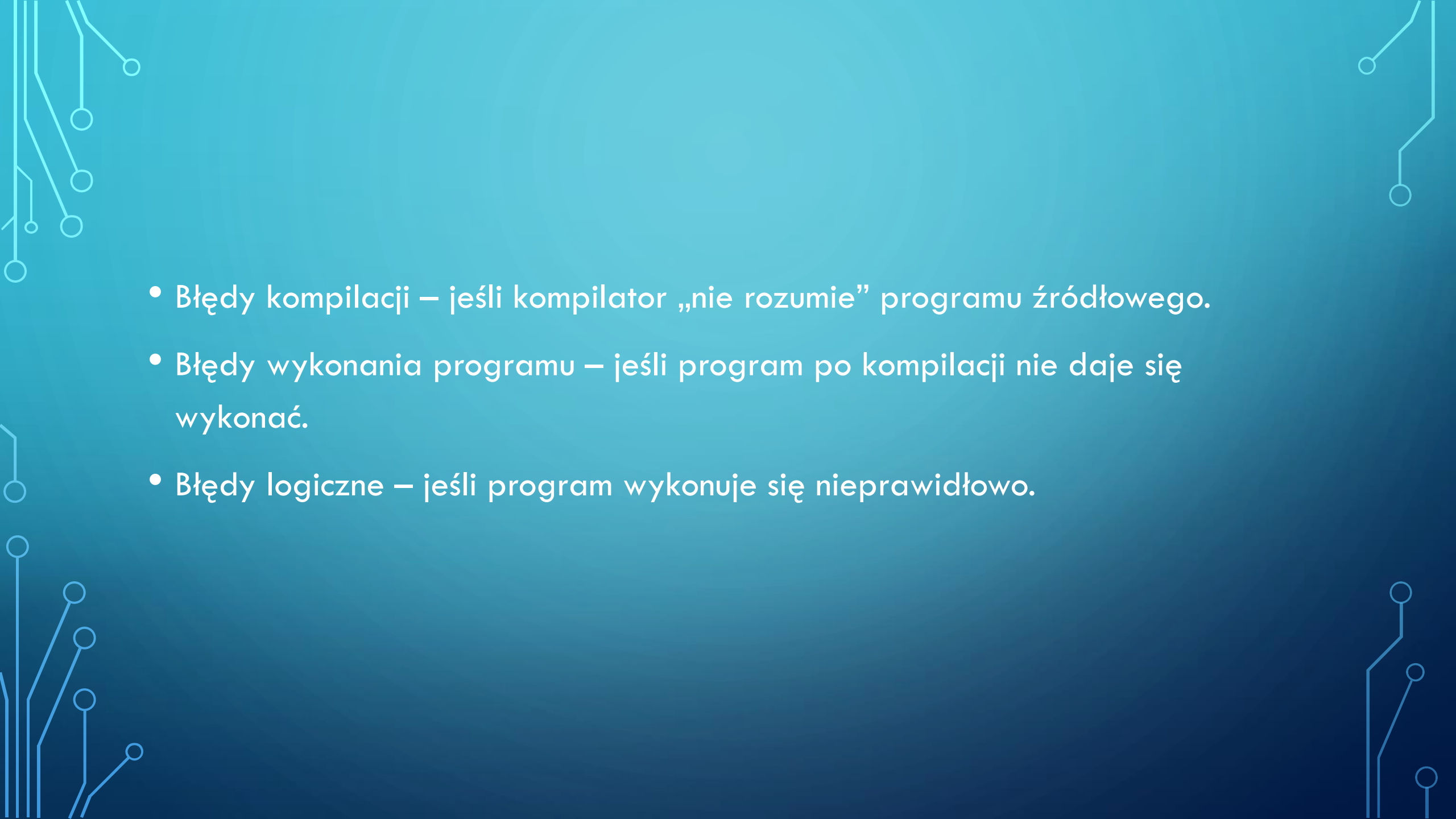
The background is a blue gradient. In the corners, there are white line-art illustrations of circuit boards or neural networks, with lines connecting to small circles.

Język programowania: **C/C++**

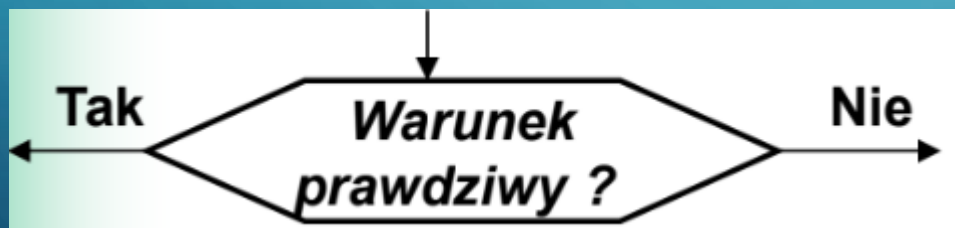
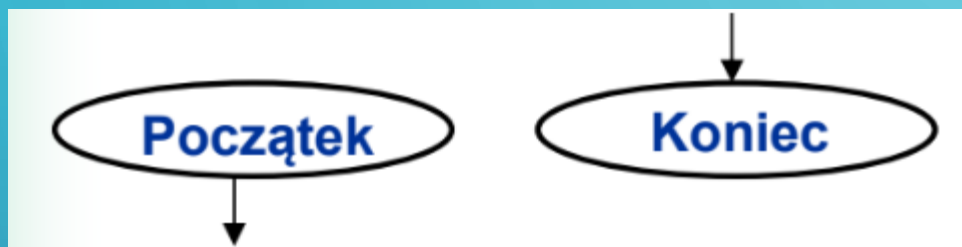
Środowisko programowania: **Visual Studio**
Arduino.ide

The background is a blue gradient. In the corners, there are white line-art illustrations of circuit boards or neural networks, with lines connecting to small circles.

PODSTAWOWE POJĘCIA, PROSTE PROGRAMY, INSTRUKCJE WARUNKOWE, PĘTLE, SORTOWANIE

- 
- The background is a solid blue gradient. In the corners, there are decorative white line art elements resembling circuit boards or neural networks, with lines and small circles connecting them.
- Błędy kompilacji – jeśli kompilator „nie rozumie” programu źródłowego.
 - Błędy wykonania programu – jeśli program po kompilacji nie daje się wykonać.
 - Błędy logiczne – jeśli program wykonuje się nieprawidłowo.

Algorytmy przedstawiane są z różnym stopniem szczegółowości. Najczęściej stosuje się: opis słowny lub schematy blokowe.



STRUKTURA PROGRAMU

include <iostream>

.....

using namespace std;

int main ()

{

*definicje, deklaracje
i instrukcje*

return 0;

}

.....

dołączanie plików nagłówkowych bibliotek

*// udostępnienie nazw ze standardowych bibliotek
// po znakach // wpisujemy komentarze ignorowane przez kompilator*

funkcja main (główna) - musi być w programie

inne funkcje - niekonieczne

Przykładowe słowa kluczowe:

ciąg znaków o ściśle określonym znaczeniu; np. (tych będziemy używać):

bool break case char const continue do else extern false float for goto if int
namespace new not or return short sizeof struct switch true using void while

Przykładowe symbole specjalne:

- jednoznakowe: + - * / = < > () [] { } . , ; : ' ^ " % ~
- dwuznakowe (dwa znaki traktowane jako całość): != <= >= == ++ -- || &&

ZMIENNE

- Są to dane, które mogą przyjmować różne wartości
- Traktujemy je jako „pudełeczka”, w których możemy coś przechowywać
- Posiadają one kilka parametrów takich jak:
 - ❖ typ (np. int, double, char, string, bool)
 - ❖ nazwę
 - ❖ miejsce w pamięci

DEFINIOWANIE ZMIENNYCH

- Definicja zmiennej Informuje o typie zmiennej i przydziela na nią pamięć.
- Najpierw podaje się typ, a potem wymienia nazwy wszystkich zmiennych tego typu.

```
typ    nazwa_1, nazwa_2, ... ;
```

- Mogą być definiowane w dowolnych miejscach w zależności od potrzeby

DEFINIOWANIE STAŁYCH

- Pewne stałe definiujemy, jeśli chcemy je nazwać i używać ich poprzez nazwy w programie.
- Stałe definiujemy w podobny sposób jak zmienne dopisując const przed typem zmiennej
- `const typ nazwa = wyrażenie;`

WYRAŻENIA ARYTMETYCZNE

- Jednoznakowe operatory:

+ - * / %

- Priorytet operatorów w wyrażeniu:

1. + oraz – oznaczający liczę dodatnią/ujemną
2. * / %
3. + -

Uwaga: W sytuacjach niepewnych najlepszym rozwiązaniem jest stosowanie nawiasów

KONWERSJA TYPÓW ZMIENNYCH

- Jest to zmiana typu zmiennej na inny
- Przykład:

Dzielenie można wykonać dla różnych typów argumentów. W razie potrzeby kompilator dokonuje niejawnego konwersji typów, np. zamienia liczbę rzeczywistą na całkowitą, obcinając ją.

UWAGA: dzielenie dwóch liczb całkowitych daje wynik całkowity (!!)

powstały przez obcięcie części ułamkowej wyniku dzielenia:

$$8/3 = 2 \quad 8.0/3 = 2.6667$$

$$-8/3 = -2 \quad -8.0/3 = -2.6667$$

OPERATORY RZUTOWANIA

- Służą one do jawnego konwertowania typów, np. `double(x)` zamienia dowolny typ zmiennej `x` na typ `double`. Dzięki temu np. możemy policzyć dokładnie iloraz `x/y` w przypadku, gdy `x` i `y` są typu `int`, pisząc: `double(x)/y` albo `(double)x/y`
- Można napisać też po prostu: `x*1.0/y`
- Uwaga: nie wolno napisać `double(x/y)`

DRUKOWANIE NA KONSOLĘ

- Instrukcja drukowania służy do wypisywania wyników standardowo na konsolę – monitor komputera
- Jest to instrukcja, która zawarta jest w bibliotece **iostream** i wymaga jej załączenia
- Biblioteka ta jest standardową biblioteką, więc aby ją załączyć wystarczy na początku programu dołączyć linię: `#include <iostream>`
- Postać instrukcji drukowania: **cout** << wyrażenie1 << wyrażenie2 << ...;
- Funkcja endl oznacza przejście do nowej linii
`cout << " Hello world" << endl;`
`cout << " Hello world \n";`
- Znaki specjalne:
`\n` `\t` `\'` `\\` `\a` `\"`

CZYTANIE Z KONSOLI

- Instrukcja czytania służy do wpisywania danych do zmiennych standardowo z konsoli – klawiatury
- Jest to instrukcja, która zawarta jest w bibliotece **iostream** i również wymaga jej załączenia
- Postać instrukcji czytania: **cin** >> zmienna1 >> zmienna2 >> ...;
- **cin.get (znak);** gdzie znak jest zmienną typu char
- **cin.get ();** - poczekaj na wczytanie jakiegoś znaku
- **getline (cin, napis);** (biblioteka string)

ZADANIE

Napisz program, który obliczy obwód koła, gdzie π zdefiniowane jest jako stała, a promień jest zmienną.

Wynik wydrukuj na konsolę.

Niech użytkownik ma możliwość wprowadzenia wartości promienia koła

INSTRUKCJA ZŁOŻONA

- Jest to zmiana ciągu instrukcji na jedną
- Wykorzystywana tam, gdzie dozwolona jest jedna instrukcja, np. w instrukcji warunkowej
- Przykład:

```
{  
a=2+x;  
a=a+1;  
cout << a;  
}
```

INSTRUKCJE WARUNKOWE

- Jest to instrukcja, która uzależnia wykonanie pewnej instrukcji od podanego warunku

- Operatory relacji logicznych:

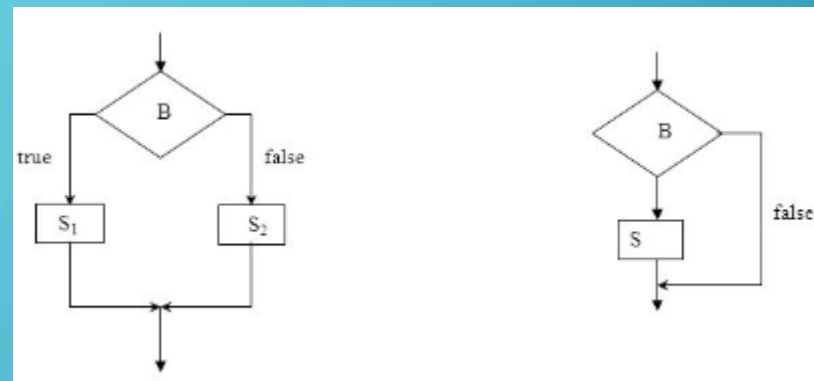
> >= < <= == !=

- Postać instrukcji:

```
if (warunek){  
    ciąg instrukcji  
}
```

```
if(warunek){  
    ciąg instrukcji  
}  
else{  
    ciąg instrukcji  
}
```

Uwaga: Instrukcje warunkowe można umieszczać w innych instrukcjach warunkowych, tworząc zagnieżdżenia



ZADANIE

Napisz program, który obliczy obwód koła, gdzie π zdefiniowane jest jako stała, a promień jest zmienną.

Wynik wydrukuj na konsolę.

Niech użytkownik ma możliwość wprowadzenia wartości promienia koła

Następnie spytaj użytkownika o pewną liczbę. Jeśli obwód jest większy lub mniejszy od tej liczby, poinformuj użytkownika

WYBRANE FUNKCJE MATEMATYCZNE

- Funkcje matematyczne inne od podstawowych jednoznakowych instrukcji typu dodaj, podziel itp. wymagają dołączenia osobnej biblioteki **cmath**

Nazwa	Argumenty	Opis
acos	X	arcus cosinus z X (wynik w radianach)
asin	X	arcus sinus z X (wynik w radianach)
atan	X	arcus tangens z X (wynik w radianach)
atan2	Y, X	kąt (od -pi do pi) , którego tangens = Y/X
ceil	X	wartość X zaokrąglona w górę, czyli do najbliższej liczby całkowitej nie mniejszej niż wartość X ceil (-4.3) = -4.0
cos	X - wartość kąta w radianach	cosinus kąta X
exp	X	wartość e podniesiona do potęgi X
fabs	X	wartość bezwzględna liczby X
log	X - wartość dodatnia	logarytm naturalny z X
log10	X - wartość dodatnia	logarytm o podstawie 10 z X
pow	X - podstawa, Y - wykładnik	X podniesione do potęgi Y pow (27, 1.0/3)=3.0
floor	X	wartość X zaokrąglona w dół, czyli do najbliższej liczby całkowitej nie większej niż X floor (-4.3) = -5.0
sin	X - wartość kąta w radianach	sinus kąta X
sqrt	X - wartość nieujemna	pierwiastek kwadratowy z X
tan	X - wartość kąta w radianach	tangens kąta X

WYRAŻENIA LOGICZNE

- Służą one do tworzenia bardziej zaawansowanych i złożonych warunków
- Wykorzystywane operatory:
|| && !
- Priorytet operatorów:
 1. !
 2. + - * / % w odpowiedniej kolejności omówionej wcześniej
 3. > >= < <=
 4. == !=
 5. &&
 6. ||

Uwaga: W sytuacjach niepewnych najlepszym rozwiązaniem jest stosowanie nawiasów

ZADANIE

Napisz program, który obliczy obwód koła, gdzie π zdefiniowane jest jako stała, a promień jest zmienną.

Wynik wydrukuj na konsolę.

Niech użytkownik ma możliwość wprowadzenia wartości promienia koła

Następnie spytaj użytkownika o pewne dwie liczby, pomiędzy którymi spodziewa się wyniku. Jeśli zgadł poinformuj go o tym.

PĘTLE

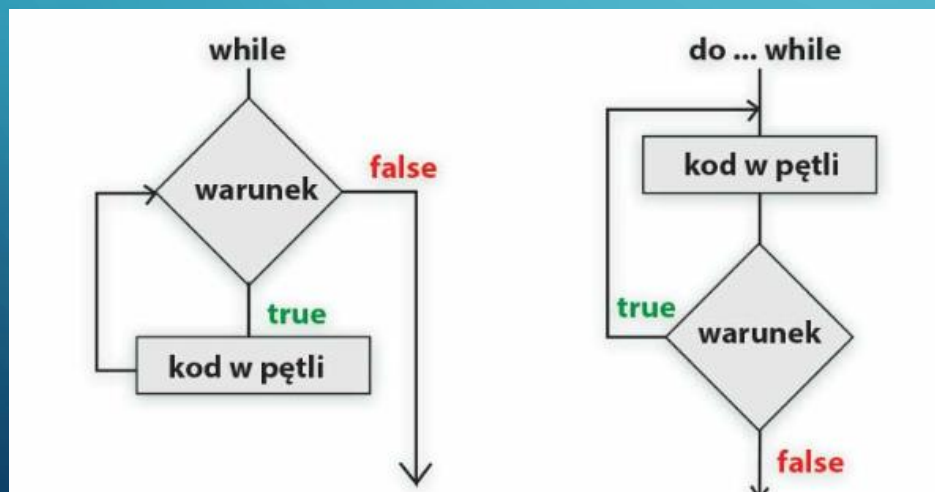
Zadanie

Napisz program który zsumuje 3 liczby podane przez użytkownika

PĘTLE

- Umożliwiają powtarzanie fragmentu kodu aż do napotkania warunku przerywającego
- Rodzaje pętli:

for



- Jaka jest różnica między pętlą while, a do-while?

ZADANIE

Program echo.

Wczytywać napis od użytkownika i wypisywać go na konsolę, aż do momentu podania przez użytkownika napisu „stop”

INKREMENTACJA I DEKREMENTACJA

- Pozwalają na skrócenie zapisów:

$a = a + 1$

$b = b - 1$

w zapisy równoważne:

$a++$

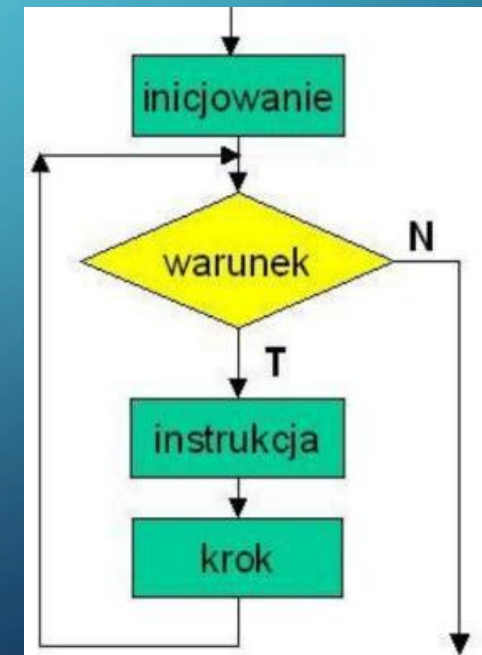
$b--$

PĘTLA FOR

- Również wykonuje się do spełnienia warunku, lecz warunkiem zazwyczaj jest określona liczba iteracji
- Postać pętli:

```
for(inicjowanie; warunek; krok){  
    instrukcje  
}
```
- Przykład:

```
for(int i = 0; i < 5; i++){  
    instrukcje;  
}
```



UWAGA: ZASIĘG ZMIENNYCH

- Zmienne, które zdefiniujemy w pewnym bloku instrukcji są dostępne w całym tym bloku oraz blokach wewnętrznych, natomiast poza nim tracimy do nich dostęp!

Zadanie

Dla wszystkich par xy liczb z przedziałów

$x = 1, 2, 3 \dots 10;$

$y = 22, 25, 28, \dots 55;$

Wydrukuj wartość x oraz y , a następnie wartość wyrażenia $7*x + 2*y$

PĘTLE ZAGNIEŹDŻONE

Zadanie

Dla wszystkich par xy liczb z przedziałów

$x = 1, 2, 3 \dots 10;$

$y = 22, 25, 28, \dots 55;$

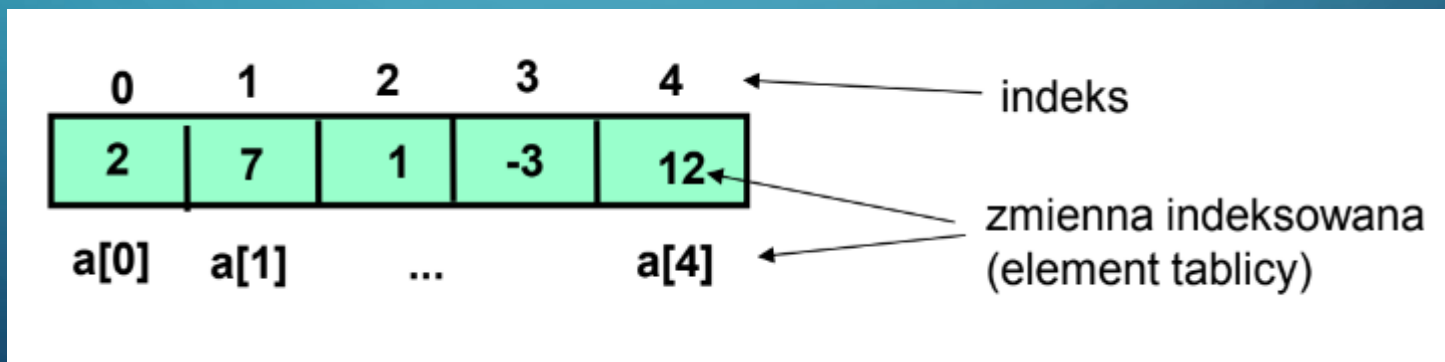
Wydrukuj wartość x oraz y , a następnie wartość wyrażenia $7*x + 2*y$

INSTRUKCJE PRZERYWAJĄCE

- Return – przerywa działanie funkcji, w tym funkcji main
- Break – przerywa działanie pętli
- Continue – powoduje przejście do kolejnej iteracji pętli

TABLICE

- Pozwalają one na zapamiętanie wiele elementów pod jedną nazwą zmiennej
- Elementy tablicy zawsze są tego samego typu
- Elementy przechowywane są jeden za drugim w pamięci
- Dostęp do konkretnych elementów realizuje się podając ich indeksy



TABLICE

- Definiowanie tablicy:

`typ_elementów Nazwa[rozmiar]`

`typ_elementów Nazwa[rozmiar] = {..., ..., , ...}`

- Uwaga: w drugim przypadku rozmiar można pominąć