



Dev-C++

## Część 3

### Algorytmika

#### Ćwiczenia wykonane pisemnie (notatki w zeszycie)

Zapisz numer pytania ( przed pytaniem np. Pytanie 1) poniżej zapisz treść pytania. Treść pytania podkreśl na **zielono** a pod treścią pytania zapisz odpowiedź.

**Pytanie 1**

Podaj definicję algorytmu;

**Pytanie 2**

Podaj metody i sposoby przedstawiania algorytmów;

**Pytanie 3**

Omów cechy charakterystyczne poprawnego algorytmu;

**Pytanie 4**

Scharakteryzuj etapy konstruowania algorytmu;

**Pytanie 5**

Omów sposoby przedstawiania algorytmów;

**Pytanie 6**

Z jakich elementów składa się specyfikacja problemu;

**Pytanie 7**

Określ co nazywamy czynnościami, danymi a co wynikami;

**Pytanie 8**

Wyясnij co zawiera program komputerowy i jakie wartości programu charakteryzuje opis obiektu;

**Pytanie 9**

Podaj rodzaje i omów znaczenie symbole stosowane w schematach blokowych;

**Pytanie 10**

Wyясnij zasady rysowania schematów blokowych;

**Pytanie 11**

Podaj i omów podział algorytmów;

**Pytanie 12**

Omów różne rodzaje złożoności programów.

**Definicja 1.**

**Algorytm** → jest to pewien ciąg czynności, który prowadzi do rozwiązania danego problemu.

**Definicja 2.**

**Algorytm** → to jednoznaczny przepis, opisujący krok po kroku sposób postępowania w celu rozwiązania pewnego problemu lub sposobu osiągnięcia jakiegoś celu.

Ilość kroków algorytmu zależy od tego, jak złożony jest problem, którego on dotyczy. Zawsze jednak liczba tych kroków będzie **liczbą skończoną**.

**Algorytmy można przedstawiać m.in. następującymi sposobami:**

- słowny opis
- schemat blokowy
- lista kroków
- drzewo algorytmu
- drzewo wyrażeń
- w pseudojęzyk
- w języku programowania.

**Sposoby przedstawiania algorytmów:**

- 1)Opis słowny (np. książka kucharska),
- 2)Schemat blokowy,
- 3)Lista kroków
- 4)program w dowolnym języku programowania.

**Cechy charakterystyczne poprawnego algorytmu:**

1. **Poprawność** - dla każdego przypisanego zestawu danych, po wykonaniu skończonej liczby czynności, algorytm prowadzi do poprawnych wyników.
2. **Jednoznaczność** - w każdym przypadku zastosowania algorytmu dla tych samych danych otrzymamy ten sam wynik.
3. **Szczegółowość** - wykonawca algorytmu musi rozumieć opisane czynności i potrafić je wykonywać.
4. **Uniwersalność** - algorytm ma służyć rozwiązywaniu pewnej grupy zadań, a nie tylko jednego zadania. Przykładowo algorytm na rozwiązywanie równań w postaci  $ax + b = 0$  ma je rozwiązać dla dowolnych współczynników  $a$  i  $b$ , a nie tylko dla jednego konkretnego zadania, np.  $2x + 6 = 0$

**Etapy konstruowania algorytmu(programu):**

1. Sformułowanie zadania.
2. Określenie danych wejściowych.
3. Określenie wyniku oraz sposobu jego prezentacji.
4. Ustalenie metody wykonania zadania.
5. Przy użyciu wybranej metody następuje zapisanie algorytmu.
6. Dokonujemy analizy poprawności rozwiązania.
7. Testowanie rozwiązania dla różnych danych.
8. Ocena skuteczności tegoż algorytmu.

**Różne sposoby przedstawiania algorytmów**

**a)opis słowny**

Jest na ogół pierwszym, mało ścisłym opisem sposobem rozwiązania problemu. Rozpoczyna się często dyskusją, w jaki sposób można rozwiązać postawione zadanie. Dyskusja służy do rozważań nad sposobem i technikami przydatnymi w rozwiązaniu problemu.

*np. Opis słowny do algorytmu opisującego funkcję modułu (wartość bezwzględna).*

Dla wartości dodatnich argumentu  $x$  funkcja przyjmuje wartość  $x$ , dla wartości ujemnych argumentu  $x$  funkcja przyjmuje wartość  $-x$ .

**b)schemat blokowy**

**c)lista kroków**

Poszczególne kroki zawierają opis operacji, które mają być wykonane przez algorytm. Mogą w nich również wystąpić polecenia związane ze zmianą kolejności wykonywanych kroków. Kolejność kroków jest wykonywana w kolejności ich opisu z wyjątkiem sytuacji gdy jedno z

poleceń w kroku jest przejściem do kroku o podanym numerze. Budowa opisu algorytmu w postaci listy kroków jest następujący:

- ♦ tytuł algorytmu
- ♦ specyfikacja problemu
- ♦ lista kroków
- ♦ komentarze ujęte w nawiasy klamrowe {komentarz}

uwaga: Krok 0 może być opuszczony

np. Lista kroków dla funkcji  $SGN(x)$  czytaj *signum*.

#### Algorytm obliczania wartości funkcji $SGN(x)$

Dane: Dowlona liczba rzeczywista  $x$ .

Wynik: Wartość funkcji

**Krok 0.** Wczytaj wartość danej  $x$

**Krok 1.** Jeśli  $x > 0$ , to  $f(x) = 1$ . Zakończ algorytm.

**Krok 2.** {W tym przypadku  $x \leq 0$ .} Jeśli  $x = 0$ , to  $f(x) = 0$ . Zakończ algorytm.

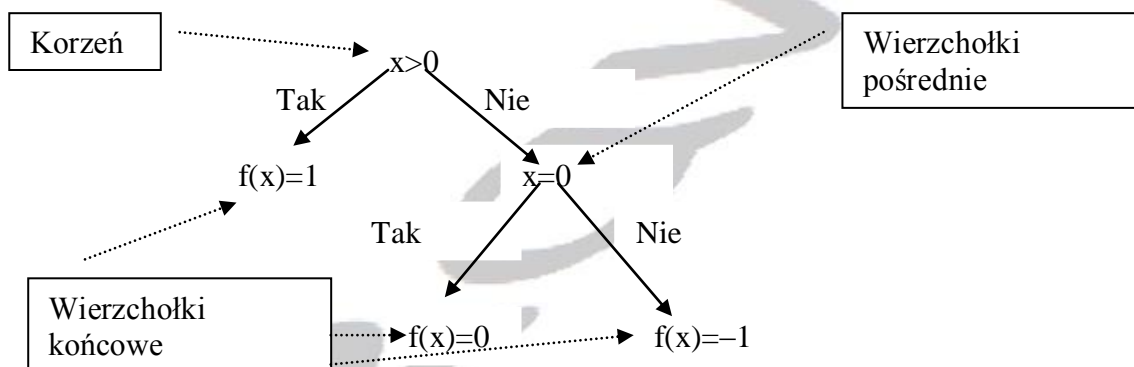
**Krok 3.** {W tym przypadku  $x < 0$ .} Mamy  $f(x) = -1$ . Zakończ algorytm.

$$SGN(x) = \begin{cases} -1 & \text{dla } x < 0 \\ 0 & \text{dla } x = 0 \\ 1 & \text{dla } x > 0 \end{cases}$$

#### d) drzewo algorytmu

Nazywany jest również drzewem obliczeń. Każde dwie drogi obliczeń mogą mieć tylko początkowe fragmenty wspólne, ale po rozejściu już się nie spotkają.

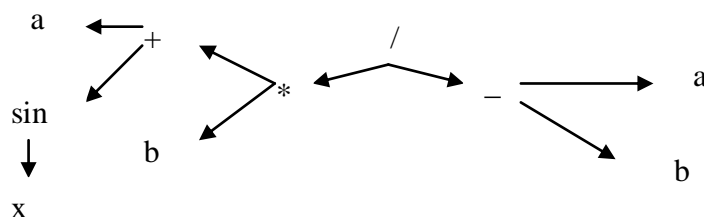
np. Drzewo algorytmu dla funkcji  $SGN(x)$ .



#### e) drzewo wyrażeń

Stosowane do obliczeń wyrażeń arytmetycznych.

np. Wyrażenie  $(a + \sin(x)) * b / (a - b)$



#### f) program w języku programowania np. Pascal

#### g) pseudojęzyk

```

PROGRAM Wycieczka;
ZMIENNE punkty:naturalne;
        koszty, dofinansowanie:rzeczywiste;
ZACZNIJ;
        WPROWADŹ(PUNKTY,KOSZTY);
        JEŚLI punkty >=100 i punkty <= TO dofinansowanie :=1/3*koszty+0.2*koszty
                W PRZECIWNYM WYPADKU
dofinansowanie:=0.2*koszty;
        WYPROWADŹ('Dofinansowanie wynosi:'dofinansowanie);
ZAKOŃCZ.

```

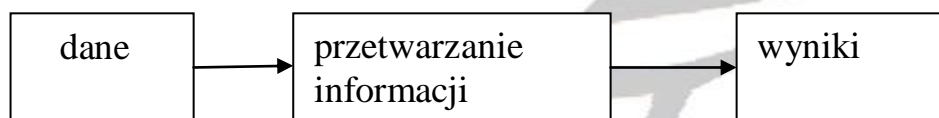
### Specyfikacja problemu

Jest to dokładny opis problemu, który chcemy rozwiązać.

Specyfikacja składa się z:

- ◆ **danych wejściowych**,
- ◆ **dane wyjściowe** oraz warunki jakie muszą spełniać (czyli związek pomiędzy danymi a wynikami).
- ◆ **warunki** jakie muszą spełniać dane wejściowe
- ◆ **rysunki** (jeśli są konieczne), **wzory** obliczeniowe

**Czynności** są wykonywane na obiektach. Obiekty podlegające przekształceniu nazywamy **danymi**. Ostateczne rezultaty wykonywania algorytmu nazywamy **wynikami**.

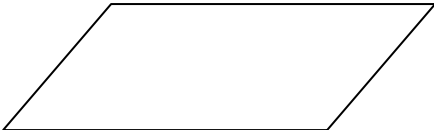
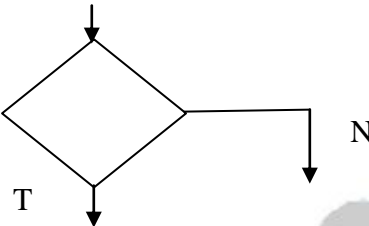

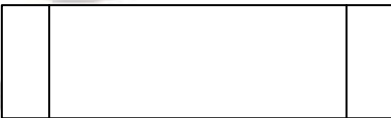





**Program komputerowy** to algorytm zakodowany w języku programowania. (języku zrozumiałym dla komputera)

**Opis obiektu** – deklaracje.

**Czynność** – instrukcja lub rozkaz.

Symbole stosowane w schematach blokowych.		
Początek algorytmu		W każdym algorytmie musi się znaleźć dokładnie jedna taka figura z napisem "Start" oznaczająca początek algorytmu. Blok symbolizujący <b>początek</b> algorytmu ma dokładnie jedną strzałkę wychodzącą
Koniec algorytmu		W każdym algorytmie musi się znaleźć dokładnie jedna figura z napisem "Stop" oznaczająca koniec algorytmu. Najczęściej popełnianym błędem w schematach blokowych jest umieszczanie kilku stanów końcowych, zależnych od sposobu zakończenia programu. Blok symbolizujący <b>koniec</b> ma co najmniej jedną strzałkę wchodzącą.

blok wyjścia		Równoległobok jest stosowany do odczytu lub zapisu danych. W jego obrębie należy umieścić stosowną instrukcję np. Read(x) lub Write(x) (można też stosować opis słowny np. "Drukuj x na ekran"). Figura ta ma dokładnie jedną strzałkę wchodzącą i jedną wychodzącą. Jest to <b>blok wyjścia/wejścia</b> wy/we I/O.
blok decyzyjny		Romb symbolizuje <b>blok decyzyjny</b> . Umieszcza się w nim jakiś warunek (np. "x>2"). Z dwóch wierzchołków rombu wyprowadzamy dwie możliwe drogi: gdy warunek jest spełniony (strzałkę wychodzącą z tego wierzchołka należy opatrzyć etykietą "Tak") oraz gdy warunek nie jest spełniony „Nie”. Każdy romb ma dokładnie jedną strzałkę wchodzącą oraz dokładnie dwie strzałki wychodzące.
blok przetwarzania		Jest to figura oznaczająca proces. W jej obrębie umieszczamy wszelkie <b>obliczenia lub podstawienia</b> . Proces ma dokładnie jedną strzałkę wchodzącą i dokładnie jedną strzałkę wychodzącą.
blok podprogramu		Ta figura symbolizuje proces, który został już kiedyś zdefiniowany. Można ją porównać do procedury, którą definiuje się raz w programie, by następnie móc ją wielokrotnie wywoływać. Warunkiem użycia jest więc wcześniejsze zdefiniowanie procesu. Podobnie jak w przypadku zwykłego procesu i tu mamy jedno wejście i jedno wyjście.
łącznik stronicowy		Koło symbolizuje tzw. łącznik stronicowy. Może się zdarzyć, że chcemy "przeskoczyć" z jednego miejsca na kartce na inne. Możemy w takim wypadku posłużyć się łącznikiem. Umieszczamy w jednym miejscu łącznik z określonym symbolem w środku (np. cyfrą, literą) i doprowadzamy do niego strzałkę. Następnie w innym miejscu kartki umieszczamy drugi łącznik z takim samym symbolem w środku i wyprowadzamy z niego strzałkę. Łączniki występują więc w parach, jeden ma tylko wejście a drugi wyjście.
łącznik		Ten symbol to łącznik międzystronicowy. Działa analogicznie jak pierwszy, lecz nie

<b>międzystronicowy</b>		w obrębie strony. Przydatne w złożonych algorytmach, które nie mieszczą się na jednej kartce.
<b>element łączący</b>		Poszczególne elementy schematu łączy się za pomocą strzałek. W większości przypadków blok ma jedną strzałkę wchodzącą i jedną wychodzącą

[www.rafalbaran.net/mb](http://www.rafalbaran.net/mb)

**Strona z magicznymi blokowymi.**

### **Reguły rysowania schematów blokowych**

- I. Po zbudowaniu schematu blokowego nie powinno być takich strzałek, które z nikąd nie wychodzą, lub do nikąd nie dochodzą.
- II. Każdy schemat blokowy musi mieć tylko jeden element startowy oraz co najmniej jeden element końca algorytmu.
- III. Element łączący(strzałki łączące) powinien być rysowany w poziomie i pionie, załamania pod kątem prostym.

### **Podział algorytmów.**

#### Definicja algorytmu liniowego

Algorytmem liniowym nazywamy taki algorytm, który ma postać listy kroków wykonywanych zgodnie z ich kolejnością.

Algorytmy liniowe są zapisem obliczeń, które mają postać ciągu operacji rachunkowych wykonywanych bez sprawdzania jakichkolwiek warunków.

#### Algorytm z warunkami (rozgałęzieniami)

Ten typ algorytmu musi mieć bloki decyzyjne czyli bloki sprawdzania warunków.

#### Algorytm numeryczne

Algorytmy, które wykonują działania matematyczne na danych liczbowych, nazywamy algorytmami numerycznymi.

#### Algorytm typu dziel i zwyciężaj

Dzielimy problem na kilka mniejszych, a te znowu dzielimy, aż ich rozwiązania staną się oczywiste,

#### Algorytmy iteracyjne

Iteracja jest to zapętlenie algorytmu, czyli wykonywania danych działań, dopóki warunek iteracji nie zostanie spełniony. Jest ona podstawą wszystkich choć troszkę bardziej złożonych algorytmów. Zazwyczaj ma ona składnię wykonuj "jakaś czynność" dopóki "jakieś wyrażenie logiczne".

#### Algorytmy rekurencyjne

Rekurencje wykorzystuje się do rozwiązywania problemów gdzie powtarza się czynność aby do niego dojść. Swoim działaniem przypomina iteracje. Jednak w tym przypadku **funkcja sama siebie wywołuje, dopóki nie otrzyma rozwiązania**, natomiast tam mieliśmy powtórzenie pewnej czynności określoną ilość razy.

**Złożoność algorytmu-** ilość zasobów potrzebnych do poprawnego działania danego algorytmu

**Złożoności obliczeniowa-**Algorytm wykonujący najmniejszą ilość operacji podstawowych w celu rozwiązania problemu.

**Złożoność czasowa-** Określa ilość operacji podstawowych potrzebnych do wykonania algorytmu o danej wielkości wejściowej.



**Złożoność pamięciowa-** Określa ilość przestrzeni pamięci wirtualnej potrzebnej do wykonania algorytmu z określonym zestawem danych wejściowych.

## Część 3A

### While, repeat, IF, Continue oraz Break

**Zadanie (praca domowa)**

**Pytanie 1**

Opisz konstrukcję pętli while, jaka to pętla, narysuj schemat, przepisuj program jako przykład.

**Pytanie 2**

Opisz instrukcję kbhit()

**Pytanie 3**

Opisz konstrukcję pętli do while, jaka to pętla, narysuj schemat, przepisuj program jako przykład.

**Pytanie 4**

Opisz konstrukcję instrukcji warunkowej if z własnym przykładem innym niż w instrukcji, narysuj schemat.

**Pytanie 5**

Opisz konstrukcję instrukcji warunkowej if...else z własnym przykładem innym niż w instrukcji, narysuj schemat.

**Pytanie 6**

Na podstawie przykładu zapisz co oznacza zapis `SUMA+=(++n);`

**Pytanie 7**

Napisz przykład zagnieżdżonej instrukcji if z własnym przykładem innym niż w instrukcji.

**Pytanie 8**

Opisz instrukcję Continue oraz Break

#### PĘTLA while.

Pętlę typu **while** jest pętlą z **kontrolowanym wejściem** tzn. najpierw jest obliczany warunek a po jego spełnieniu wchodzimy do pętli i wykonujemy instrukcje z niej.

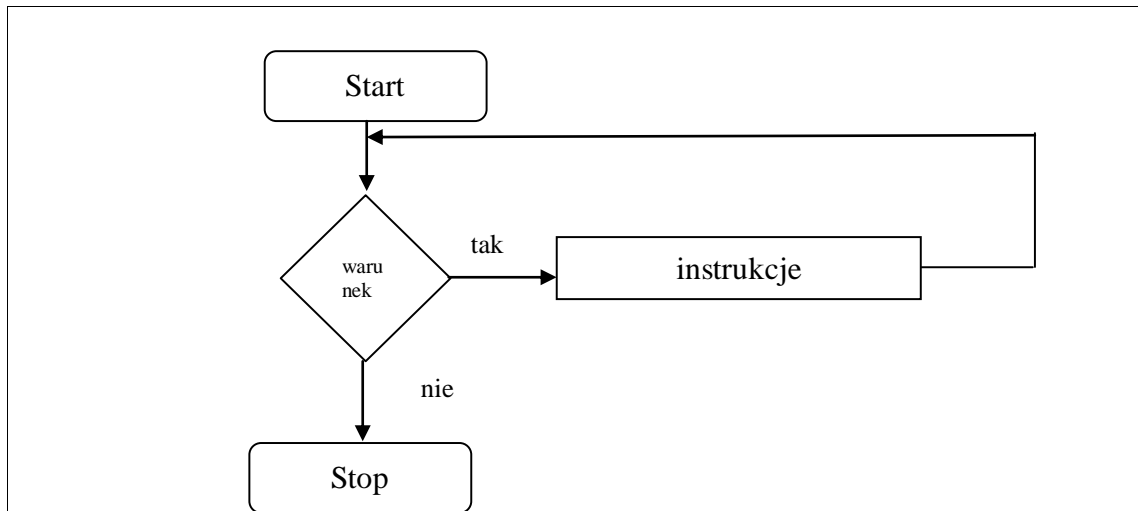
(najpierw sprawdza potem wykonuje!)

Konstrukcja pętli while wygląda następująco:

```
while (Wyrażenie_logiczne)
```

```
{
    Instrukcja1;
    .....
    InstrukcjaN;
}
```

Jeśli Wyrażenie\_logiczne ma wartość logiczną zera, to nie zostaną wykonane Instrukcje czyli nie nastąpi wejście do pętli..



### Przykład 10

Temat: Wykorzystanie pętli while do programu piszącego wykrzykniki aż do wciśnięcia dowolnego klawisza.

Funkcja kbhit() oznacza wciśnięcie dowolnego klawisza.

Wykonaj:

- 1)Przepisz temat.
- 2)Zapisz teorię+schemat blokowy dotyczącą pętli while.
- 3)Zapisz, co oznacza funkcja kbhit() , !kbhit(), printf("\n"), (zwróć uwagę na wyjaśnienie na stronie 35 niniejszej instrukcji)
- 4)Uruchom przykład,
- 5)Zapisz przykład do zeszytu (pamiętaj o wcięciach).

```

#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <conio.h> // funkcja potrzebna do działania kbhit()

using namespace std;

int main(int argc, char *argv[])
{
    while(!kbhit())
    {
        printf("!");
    }
    printf("\n");
    system("PAUSE");
    return EXIT_SUCCESS;
}
  
```



**Zadanie 15a**

Uzupełnij tabelkę

Zapis tradycyjny	Zapis skrócony pierwszy	Zapis skrócony drugi
zm = zm + zm1	zm+=zm1	
?	?	p++
?	zmie+= 5	
?	ujka - = 1	-?
jj = jj % 2	?	
?	u*=10	
hitrus= hitrus/4	?	

**Inkrementacja**

Operator inkrementacji (++), czyli zwiększenie wartości zmiennej o stałą wartość najczęściej o jeden (1). Np. i=i+1; lub w formie skróconej i++;

**Dekrementacja**

Operator dekrementacji (--), czyli zmniejszenie wartości zmiennej o stałą wartość najczęściej o jeden (1). Np. x=x-1; lub w formie skróconej x--;

**Formy zapisu:**

**Przedrostkowa** np. (++i --x) najpierw zmienna jest zwiększana o jeden, a następnie ta zwiększona wartość jest brana do obliczeń.

**Przyrostkowa** (końcówkowa) (i++ x--) najpierw brana jest stara wartość zmiennej do obliczeń a dopiero później jest ona zwiększana o jeden.

**Przykład 11**

**Temat:** Stosujemy pętlę while w programie obliczającym sumę kolejnych liczb naturalnych dopóki SUMA nie przekroczy 1000.

Wykonaj:

- 1)Przepisz temat,
- 2)Zapisz, co oznacza instrukcja SUMA+=(++n);,
- 3)Uruchom przykład,
- 4)Zapisz w zeszycie daną wyjścia dla programu,
- 5)Zapisz przykład do zeszytu (pamiętaj o wcięciach).
- 6)Wykonaj schematy blokowy.

```

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float SUMA=0, n=0;
    while (SUMA<1000)
    {
        SUMA+=(++n);    // SUMA = SUMA + n;
        // ++n zwiększaj zmienną a o jeden
        //w formie przedrostkowej
        // inny zapis ++n to    n=n+1;
    }
    printf("SUMA: %.1f ostatnia liczba: %.2f", SUMA, n);
    cout<<"\n"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

**SUMA+** działa przyrostkowo czyli najpierw brana jest stara wartość zmiennej do obliczeń a dopiero później jest ona zwiększana o jeden. ( pętla while liczy od wartości zmiennej SUMA=0);

**++n** działa przedrostkowo czyli najpierw zmienna jest zwiększana o jeden, a następnie ta zwiększona wartość jest brana do obliczeń. ( pętla while liczy od wartości zmiennej n=0+1=1);

```
SUMA: 1035.0 ostatnia liczba: 45.00
```

```
Aby kontynuować, naciśnij dowolny klawisz . . .
```

### Zadanie 15b

#### Temat:

Oblicz z użyciem pętli **while** **iloczyn** liczb:

Od → [(liczba\_liter\_imienia) mod 4]+1 Do → [(liczba\_liter\_imienia) mod 4]+6

Dokonaj obliczenia i zapisz w zeszycie:

Od → *tutaj wpisz w zeszycie obliczoną liczbę*

Do → *tutaj wpisz w zeszycie obliczoną liczbę*

#### Uwaga1:

Program może sam obliczać mod lub sam możesz obliczyć wartość Od Do

#### Uwaga2:

Zmienna w, której zapisujesz wartość iloczynu to ILOCZYN\_trzy\_pierwsze\_litery\_nazwiska  
np. ILOCZYN\_kow

Wykonaj schematy blokowy.

Sprawdzenie dla imienia Andrzej:

przyjmijmy przykładowe zmienne:

n - liczba od której zaczynamy pętlę;

Koniec – liczba na której kończymy pętlę;

Iloczyn – zmienna przechowująca wartość iloczynu liczb

n		4	5	6	7	8	9	10
Iloczyn	1	4	20	120	840	6720	60480	
Koniec		9						9<10

```

C:\> D:\Andrzej\Praca\Lekcje\CPP\Cwiczenia_C++\C++Cz_3\Zadania\Zadanie_15B_C++\Zadanie_
Podaj liczbę liter Twojego imienia: 7
n=4
koniec=9
Iloczyn wynosi: 60480 ostatnia liczba: 10
Aby kontynuować, naciśnij dowolny klawisz . . . _
  
```

### PĘTLA do...while.

Pętlę typu do while jest pętlą z kontrolowanym wyjściem tzn. warunek obliczany po wykonaniu pętli.

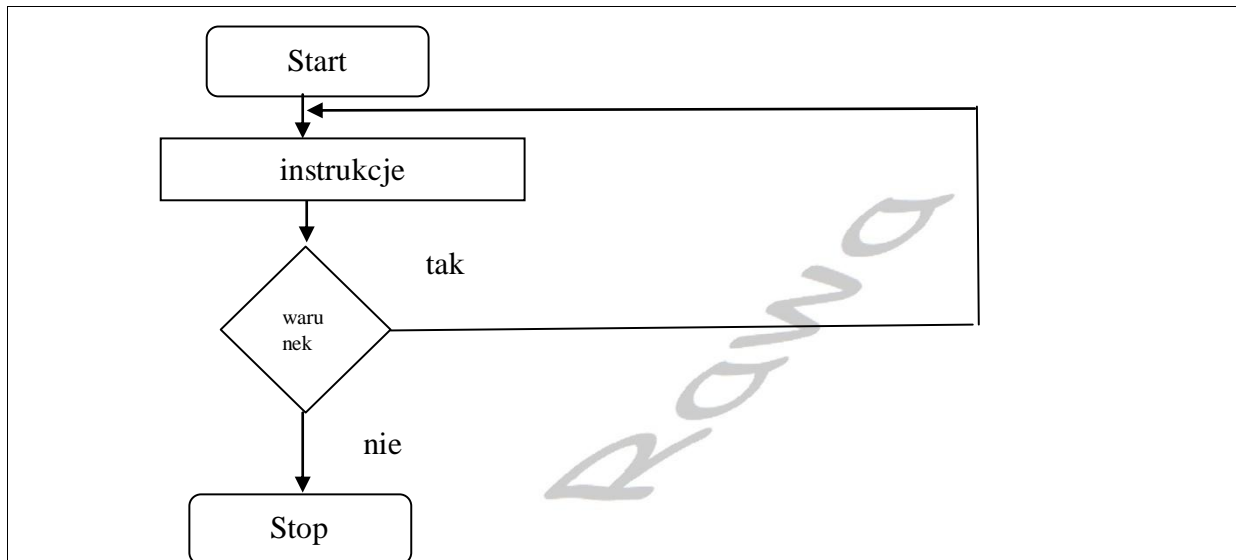
Przerwanie pętli powodowane jest przez **NIESPEŁNIENIE WARUNKU**.

Konstrukcja pętli do while wygląda następująco:

```

do
{
    Instrukcja1;
    Instrukcja2;
    .....
    InstrukcjaN;
}
  
```

while (Wyrażenie\_logiczne);



### Przykład 12

Temat: Program obliczający pole trójkąta ze sprawdzeniem poprawności wczytywania danych z użyciem pętli **do...while**.

Wykonaj:

- 1)Przepisz temat,
- 2)Uruchom przykład,
- 3)Zapisz teorię +schemat blokowy dotyczącą pętli **do .....while**.
- 4)Zapisz w zeszycie, jakich wartości **nie** przyjmuje przykład jako danych wejściowych,
- 5)Zapisz do zeszytu (pamiętaj o wcięciach) tę część programu, która sprawdza poprawność wczytywanej podstawy (pamiętaj o wcięciach).
- 6) zapisz do zeszytu

pętla **do...while** → to pętla z .....

pętla **while** → to pętla z .....

```

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    float bok_a, wysokosc, pole;
    do
    {
        cout<<"podaj dlugosc podstawy a=";
        cin>> bok_a;
    }
    while (bok_a<=0);
    do
    {
        cout<<"podaj dlugosc wysokosci=";
        cin>> wysokosc;
    }
  
```

```

    }
    while (wysokosc<=0);
    pole = bok_a*wysokosc/2;
    cout<<"pole="<<pole<<" m^2";
    cout<<"\n"<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

Koniec przykładu 12

### **Przykład 12a**

Temat: Demonstracja pętli **do...while** do zapewnienia powtarzalności programu.

Wykonaj:

- 1)Przepisz temat,
- 2)Uruchom przykład,
- 3)Zapisz przykład do zeszytu (pamiętaj o wcięciach).
- 4)Dopisz, aby powtarzanie było również na klawisz „T” użyj operatora OR ( sprawdź jak go zapisujemy w CPP) .

```

#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <conio.h>

using namespace std;

int main(int argc, char *argv[])
{
    float liczba;
    char znak;
    system ("chcp 1250");
    system ("cls");
    do
    {
        printf("Umiem powtarzać program\n");
        printf("\njeśli powtórzyć program wcisnij t(tak)\nw przeciwnym wypadku wcisnij
klawisz n(nie)\n");
        cin>>znak;
    }
    while (znak=='t');

    return EXIT_SUCCESS;
}

```

**UWAGA:** dwa znaki równości == oznaczają RÓWNA SIĘ, w przeciwieństwie do jednego znaku równości =, który oznaczał PODSTAW.

**Zadanie 16**

**Temat:** Napisz program znajdujący największy wspólny dzielnik dla nieujemnych liczb całkowitych  $m$  i  $n$  algorytmem Euklidesa. Największy wspólny dzielnik oznaczamy symbolem  $NWD(m,n)$ .

**Wykonaj:**

- 1) Zapisz temat zadania.
- 2) Zapisz w zeszycie specyfikację dla problemu znajdowania NWD algorytmem Euklidesa.
- 3) Zapisz w zeszycie listę kroków dla problemu znajdowania NWD algorytmem Euklidesa.

**Algorytm Euklidesa** – pierwszy znany **algorytm** wyznaczania **największego wspólnego dzielnika** dwóch **liczb naturalnych**. Został opisany przez greckiego matematyka **Euklidesa**.

Największy wspólny dzielnik liczb  $a$  i  $b$  zapisuje się zwykle  $nwd(a, b)$  lub  $NWD(a, b)$ , a czasem nawet  $(a, b)$ . Przykładowo  $nwd(8, 12) = 4$  oraz  $nwd(-4, 14) = 2$ . Dwie liczby nazywa się **względnie pierwszymi**, jeżeli ich największym wspólnym dzielnikiem jest  $1$ , względnie pierwsze są np.  $9$  i  $28$ .

Pojęcie największego wspólnego dzielnika wykorzystuje się podczas redukcji **ułamków** do postaci nieskracalnej (tzn. takiej, w której licznik i mianownik są względnie pierwsze). Przykładowo największym wspólnym dzielnikiem liczb  $42$  oraz  $56$  jest  $14$ , stąd

$$\frac{42}{56} = \frac{3 \cdot \cancel{14}}{4 \cdot \cancel{14}} = \frac{3}{4}.$$

Specyfikacja dla algorytmu Euklidesa.

**Dane:** Dwie liczby naturalne  $m$  i  $n$ ,  $m \leq n$ .

**Wynik:**  $NWD(m,n)$  – największy wspólny dzielnik  $m$  i  $n$

Lista kroków dla algorytmu Euklidesa.

**Krok 1** Jeśli  $m=0$ , to  $n$  jest szukany dzielnikiem. Zakończ algorytm.

**Krok 2**  $r=(n \bmod m)$ ,  $n=m$ ,  $m=r$ . Wróć do kroku 1

4) Wykonaj symulację znajdowania  $NWD(14,21)$  w postaci uzupełnionej tabeli.

przebieg algorytmu	n	m	r
1	?	?	?
.....	.....	.....	.....
.....	.....	.....	.....

```

D:\Andrzej\Praca\Lekcje\CPP\Cwiczenia_C++\Cz_3\Zadania\Zadanie_16_C++\Zadanie_16_C++.ex
podaj pierwsza liczbe n= 21
podaj druga liczbe mniejsza lub rowna pierwszej m= 14
NWD(14,21)=7
Aby kontynuować, naciśnij dowolny klawisz . . .
  
```

5) Wykonaj symulację znajdowania NWD(1073,1517) w postaci uzupełnionej tabeli.

przejścia algorytmu	n	m	r
1	?	?	?
.....	.....	.....	.....
	.	.	.
.....	.....	.....	.....
	.	.	.

```

podaj pierwsza liczbe n= 1517
podaj druga liczbe mniejsza lub rowna pierwszej m= 1073
NWD(1073,1517)=37

Aby kontynuowac, naciśnij dowolny klawisz . . .

```

6) Zapisz algorytm w postaci schemat blokowy.

7) Wykonaj program.

### Zadanie 16a

Temat; Rozwiązanie zadania maturalnego. Użycie pętli while do rozwiązywania problemu.

Wykonaj:

- 1) Przepisz temat,
- 2) Przepisz specyfikację oraz listę kroków dla tego problemu dla zadania 16a)

#### Specyfikacja algorytmu:

Dane: **N** – liczba całkowita większa od 0.

Wynik: wartość zmiennej **wyn**

#### Algorytm w postaci listy kroków:

Krok 0: wczytaj **N**

Krok 1: **wyn** := 0; **d** := 2;

Krok 2: Dopóki  $d \leq (N \text{ div } 2)$  wykonuj kroki 2.1 i 2.2;

Krok 2.1: Jeżeli  $N \bmod d = 0$ , to **wyn** := **wyn** + 1;

Krok 2.2: **d** := **d** + 1;

Krok 3: wypisz na ekranie **wyn**

*Koniec listy kroków.*

UWAGA1: „ $N \bmod d$ ” – jest równe reszcie z dzielenia całkowitego liczby **N** przez **d**,  
np.  $10 \bmod 5 = 0$ ,  $10 \bmod 3 = 1$ .

„ $N \text{ div } 2$ ” – jest równe wynikowi dzielenia całkowitego liczby **N** przez **d**,  
np.  $10 \text{ div } 5 = 2$ ,  $10 \text{ div } 3 = 3$ .

„:=” – oznacza instrukcję przypisania.

#### Uwaga2:

Instrukcja **div** w cpp jest realizowana jako zwykłe dzielenie, ale zmienna, do której zapisujemy wynik musi być **int**. Instrukcja **mod** w cpp jest realizowana jako %

#### **Wykonaj ciąg dalszy:**

3) Zapisz w zeszycie jak wykonywana jest instrukcja **div** i **mod** w CPP,

4) Uruchom program i dokonaj kompilacji wg listy kroków podanej powyżej.



Użyj:

→ `do .....while,`

→ wszystkie zmienne naturalne,

→ wyrażenia `wyn := wyn + 1` `d := d + 1` zapisz z użyciem **inkrementacji**, takiej jak używa się w CPP

→ instrukcja `if` pisze się z małej litery i porównanie jest z użyciem `==` (dwa znaki =)

Instrukcja `if` → strona 17 instrukcji `cpp_dev_Cz_3`

5) Przy użyciu programu uzupełnij tabelę, zapisz ją w zeszycie:

N → dana wejściowa	Wyn → dana wyjściowa
10	2
90	10
17	0
(Numer w dzienniku)*10	Uzupełnij sam
Dowolna Liczba pierwsza większa od 100+nr_dziennika	Uzupełnij sam

6) program zapisz w zeszycie

Koniec zadania

```

podaj wartość N: 10
wyn=2
d=6

Aby kontynuować, naciśnij dowolny klawisz . . .

```

## INSTRUKCJA WARUNKOWA `if`

Instrukcja warunkowa ma postać:

`if (Wyrażenie)`

```

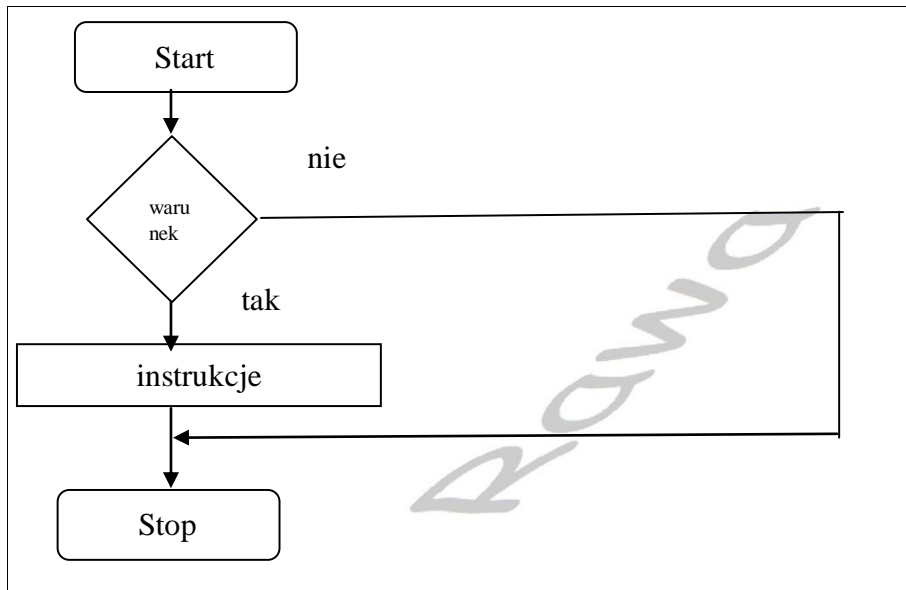
{
    Instrukcja1;
    .....
    InstrukcjaN;
}

```

np.

```
if (a<b) { cout<<"Liczba a jest mniejsza od b"<<endl;}
```

Jeśli warunek jest prawdziwy, wykonaj instrukcję lub grupę instrukcji. Gdy warunek jest fałszywy, nie rób nic.



### Przykład 13

Temat: Prezentacja instrukcji **if** dla podania komunikatu o nieprawidłowego wczytania danych.

Wykonaj:

- 1) Przepisz temat.
- 2) Zapisz teorię + schemat blokowy dotyczącą instrukcji if.
- 3) Uruchom przykład
- 4) Wpisz przykład do zeszytu.

```

#include <cstdlib>
#include <iostream>

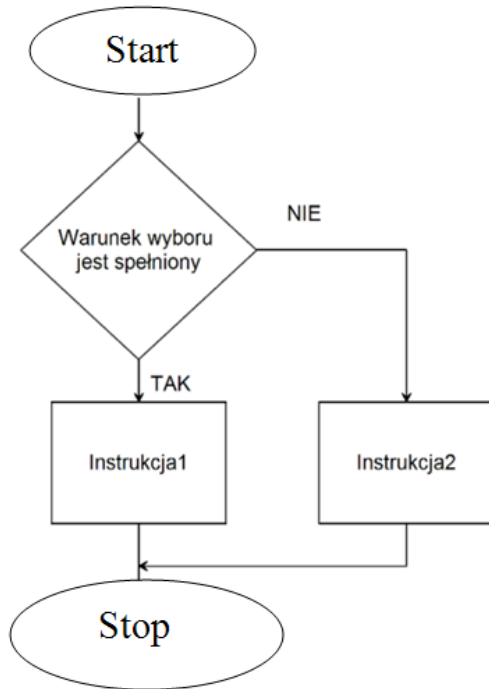
using namespace std;

int main(int argc, char *argv[])
{
    system("chcp 1250");
    system("cls");
    int waga;
    do
    {
        printf("Podaj Twoja wagę w [kg]= ");
        scanf("%d",&waga);
        if (waga<=0)
        { printf("waga musi być większa od zera\n"); }
    }
    while(!(waga>0));
    printf("waga wczytana poprawnie\n");
    system("PAUSE");
    return EXIT_SUCCESS;
}
  
```

## INSTRUKCJA WARUNKOWA if .....else.....

Instrukcja warunkowa ma postać:

if (Wyrażenie) {Instrukcja1; ...Instrukcja N;} else {Instrukcja2; ...Instrukcja N;}



np.  
 if (dzielnik !=0)  
 {  
 cout<<"można dzielic"<<endl;  
 }  
 else  
 {  
 cout<<"pamiętaj cholero nie dziel przez zero"<<endl;  
 }

Jeśli Wyrażenie jest prawdziwy to zostanie wykonana Instrukcja1, w przeciwnym razie wykonana zostanie Instrukcja2. Instrukcje mogą być instrukcjami grupującymi. Słowa kluczowe if i else mogą być stosowane wielokrotnie. Pozwala to tworzyć np. **tw. zagnieżdżenia**.

### Przykład instrukcji zagnieżdżonych:

```
if (a>0) if (a<100) printf("Dwucyfrowa"); else printf("100+");
```

inaczej:

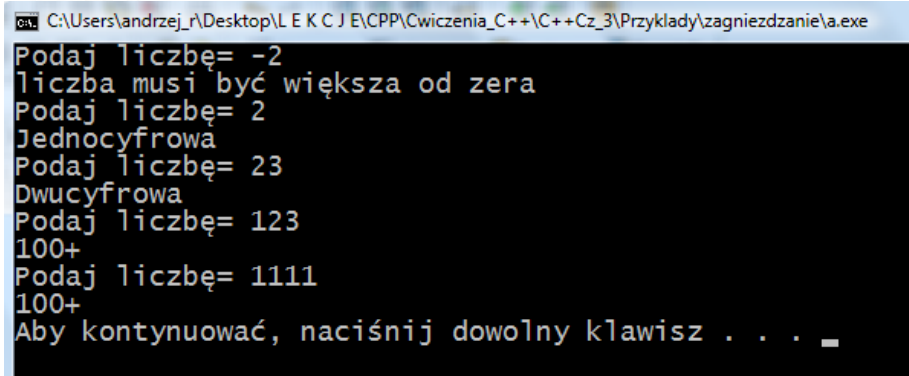
```
if(a>0) {if(a<100) printf("Dwucyfrowa"); else printf("100+");}
```

Zapis 100+ oznacza "sto i więcej".

1. zastanów się jak działa powyższy przykład, czy nie ma błędu w zapisie ?

wgraj kod do dev-a i dokonaj poprawy.

2. na podstawie powyższej instrukcji napisz program, który będzie sprawdzał ile cyfrową liczbę wprowadzono z klawiatury oraz wydawał stosowne komunikaty. Program zbuduj w pętli, która będzie miała swój koniec w przypadku wprowadzenia liczby minimum czterocyfrowej. Program powinien również podać stosowny komunikat w przypadku wprowadzenia liczby mniejszej od zera.



```

C:\Users\andrzej\Desktop\LEKCJE\CPP\Cwiczenia_C++\C++Cz_3\Przyklady\zagniezdzenie\1a.exe
Podaj liczbę= -2
liczba musi być większa od zera
Podaj liczbę= 2
Jednocyfrowa
Podaj liczbę= 23
Dwucyfrowa
Podaj liczbę= 123
100+
Podaj liczbę= 1111
100+
Aby kontynuować, naciśnij dowolny klawisz . . . _
  
```

### Przykład 12B

W nawiązując do przykładu 12A, warto zapamiętać, że istnieje inny sposób przypisać wartość wciśniętego klawisza do zmiennej typu char, będzie to bardzo pomocne: `znak=getch();`

```

#include <cstdlib>
#include <iostream>
#include <conio.h>
#include <stdio.h>
using namespace std;

int main(int argc, char *argv[])
{
    char znak; // zmienna typu char o nazwie znak

    system("chcp 1250");
    system("cls");

    cout<<"Wcisnij dowolny klawisz: "<<endl; // prośba o wciśnięcie klawisza
    znak=getch(); // za zmienną znak podstawiamy wartość zwracaną przez getch()
    cout<<"Zostal wcisniety klawisz "<<znak<<endl; // wypisujemy wartość zmiennej znak

    if(znak=='a')
    {
        cout<<"Pelny dostep"<<endl;
    }
    else
    {
        cout<<"Brak dostepu"<<endl;
    }
    getch();
}
  
```

**UWAGA:** Wszystkie znaki zapisujemy w cudzysłowie ' '.

### Zadanie 17

Temat: Użycie instrukcji **if .....else.....** oraz operatorów **AND** oraz **OR**.

Wykonaj:

- 1) Przepisz temat,
- 2) Zapisz jak tworzone są w CPP operatory **AND** oraz **OR**
- 3) Zapisz teorię + schemat blokowy dotyczącą instrukcji **if.....else.....**
- 4) Zapisz zadanie do zeszytu (pamiętaj o wcięciach).

#### Treść zadania

Napisz program z użyciem **if ....else...** (jednego) oraz operatorów **AND** lub **OR**

a) Napisze „Będę sprawdzał czy liczba należy do przedziału  $<-5,6)$ ”

b) Będzie się pytał o liczbę wczytaną do zmiennej z trzema literami nazwiska np. **x\_kow**.

c) następnie wypisywał jeden z komunikatów:

liczba należy do przedziału  $<-5,6)$  lub liczba nie należy do przedziału  $<-5,6)$

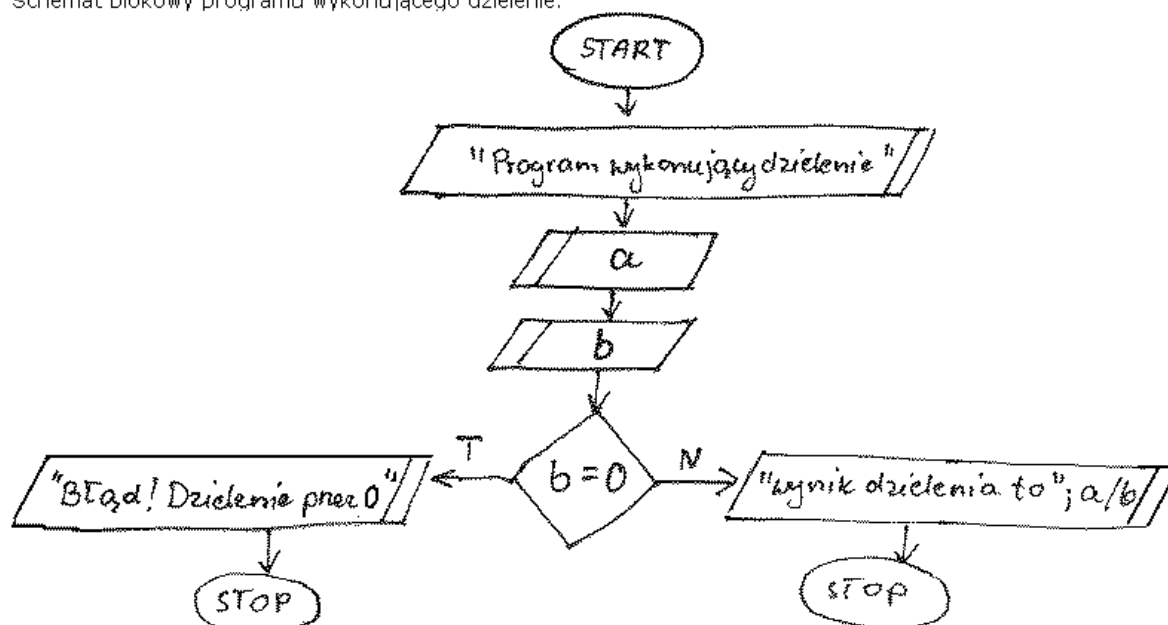
```
Będę sprawdzał czy liczba należy do przedziału <-5,6)
Podaj liczbę do sprawdzenia: -9

liczba nie należy do przedziału <-5,6)
Aby kontynuować, naciśnij dowolny klawisz . . . .
```

### Zadanie 18

Napisz program wg. poniższego algorytmu. Sprawdź czy w algorytmie występuje błąd?

Sprawdź jak program działa dla różnych wartości, spróbuj wpisać 0 jako drugą liczbę.  
Schemat blokowy programu wykonującego dzielenie:



```

Program wykonujący dzielenie
Podaj pierwszą liczbę: 4
Podaj drugą liczbę: 0
Błąd dzielenie przez 0!
Aby kontynuować, naciśnij dowolny klawisz . . .

```

### Zadanie 19

Temat: Program rozpoznaje znak liczby z klawiatury z zastosowaniem **if...else** zagnieżdżonej.

Treść zadania: Komputer pyta się o liczbę i rozpoznaje znak liczby możliwe odpowiedzi programu:

- Liczba większa od zera
- Liczba mniejsza od zera
- Liczba równa zero

Użyj dwóch instrukcji if else.

```

Podaj liczbę całkowitą: 0
liczba równa zero
Aby kontynuować, naciśnij dowolny klawisz . . .

```

### Przykład 14

Napisz program, który sprawdza podzielność jednej liczby przez drugą. (Jeżeli reszta z dzielenia równa się 0 to liczba pierwsza jest podzielna przez drugą.

Znajdź i popraw błędy w kodzie programu!

```

#include <cstdlib>
#include <iostream>
#include <conio.h> // instrukcja potrzebna do poprawnego działania getch()

using namespace std;

int main(int argc, char *argv[])
{
    system("cls");
    system("chcp 1250");

    int liczba1, liczba2, wynik;

    cout<<"Podaj pierwsza liczbę: ";
    cin<<liczba1;
    cout<<"\nPodaj druga liczbę: ";
    cin<<liczba2;

    wynik=liczba1%liczba2;

    cout<<"\nReszta z dzielenia liczby "<<liczba1<<" przez "<<liczba2<<" wynosi "<<wynik;

    if(wynik==0) then//sprawdzenie warunku

```

```

    { cout<<"\n\npierwsza liczba jest podzielna przez druga"<<endl;}; //instrukcja nr 1 - BŁĄD
    !!!
    else //jeżeli nie
    { cout<<"\n\npierwsza liczba nie jest podzielna przez druga"<<endl;}; //instrukcja nr 2 -
    BŁĄD !!!

    cout<<endl;
    getch();
}

```

**UWAGA:** w nawiasach klamrowych wpisujemy dowolną ilość instrukcji oddzielonych średnikami. Po nawiasach klamrowych NIE dajemy średnika;

```

Podaj pierwsza liczbe: 7
Podaj druga liczbe: 4
Reszta z dzielenia liczby 7 przez 4 wynosi 3
pierwsza liczba nie jest podzielna przez druga

```

#### Przykład 15

Napisz program, który udostępni nam nasze dane, gdy wpisemy poprawny kod PIN. Na początku założymy, że poprawnym kodem PIN będzie: 1753. Należy zdefiniować stałą pin oraz zmienną podany\_pin, do której zostanie zapisany kod podany przez użytkownika. Kolejnym krokiem będzie porównanie wartości zmiennych podany\_pin z pin i wypisanie odpowiedniego komunikatu.

```

#include <cstdlib>
#include <iostream>
#include <conio.h> // instrukcja potrzebna do poprawnego działania getch()

using namespace std;

int main(int argc, char *argv[])
{
    const int pin=1753; //pin jest liczbą całkowitą, jej wartość to 1753
    int podany_pin; //podany pin jest również liczbą całkowitą ;- )

    cout<<"Proszę podać czterocyfrowy kod PIN: "<<endl; //prosimy o podanie kodu PIN
    cin>>podany_pin; //wczytujemy kod z klawiatury
    cout<<"Podany kod PIN to: "<<podany_pin<<endl; //dla wprawy, potwierdzamy

    if(podany_pin==pin) //sprawdzamy warunek: jeżeli podany_pin RÓWNA SIĘ pin
    { //to
        cout<<"Kod PIN jest poprawny"<<endl; //tu mamy blok instrukcji, gdy pin jest poprawny
        cout<<"Oto Twoje dane:"<<endl<<endl;
        cout<<"Jan Kowalski"<<endl;
        cout<<"Kominki 56"<<endl;
        cout<<"34-683 Kielce"<<endl<<endl;
    }
}

```



```

cout<<"jankowalski@domena.pl"<<endl<<endl;
cout<<"Dowolny klawisz, aby zakonczyc"<<endl;
}
else //jeżeli natomiast kod PIN jest błędny to wtedy:
{
    cout<<"kod PIN nie jest poprawny"<<endl;
    cout<<"Dowolny klawisz, aby zakonczyc"<<endl;
}

getch();
}

```

### Zadanie 20

Temat: Napisać program obliczający pole kwadratu.

Sprawdź czy bok kwadratu jest większy od zera. Jeżeli jest większy oblicz pole jeśli nie to podaj komunikat "bok musi być większy od zera". W programie użyj **tylko jednej** instrukcji IF...THEN...ELSE. Zapewnij powtarzalność programu jak zapewnić powtarzalność patrz przykład poprzedni..

Wykonaj w zeszycie schemat blokowy dla zadania.

```

Podaj bok kwadratu w centymetrach: 5
Pole kwadratu =25 cm^2
jeśli powtórzyć program wcisnij t(tak)
w przeciwnym wypadku wcisnij dowolny klawisz literowy
t
Podaj bok kwadratu w centymetrach: 0
bok nie może być równy zero
jeśli powtórzyć program wcisnij t(tak)
w przeciwnym wypadku wcisnij dowolny klawisz literowy
n
Aby kontynuować, naciśnij dowolny klawisz . . . _

```

### Zadanie 21

Temat: Program obliczający wartość funkcji danej wzorem w postaci klamkowej.

Wykonaj:

- 1)Przepisz temat,
- 2)Rozwiąż zadanie

Zmienne z trzema literami nazwiska.

Napisz program obliczający wartość funkcji danej wzorem

$$f(x) = \begin{cases} 1 & \text{dla } x \geq 1 \\ x * x & \text{dla } -1 < x < 1 \\ 1 & \text{dla } x \leq -1 \end{cases}$$

Dopisz do programu jego powtarzanie czyli po podaniu  $x$  i obliczeniu  $f(x)$  komputer spyta się czy ma powtórzyć program (jak to zrobić patrz przykład poprzedni) i na wczytaną literę pierwszą litera Twojego imienia np. Jan będzie to litera „J” lub „j” nastąpi ponowne pytanie o  $x$  i obliczenie  $f(x)$ .

3) Wykonaj w zeszycie tabelę i uzupełnij ja.

X						0					
Y											

Uzupełnij pierwszy wiersz liczbami dwie  $> 1$ , dwie  $< -1$ , trzy z przedziału  $(-1,0)$ , trzy z przedziału  $(0,1)$

Pisz  $x$  a komputer obliczy  $y$ . Zapisz  $x$  i  $y$  tabeli. Uzupełnij całą tabelę.

4) Posługując się programem wykonaj wykres funkcji  $f(x)$  w zeszycie, 5 cm na osi X i Y to jeden( czyli duża skala).

Punkty z tabeli zaznacz w układzie XY. Połącz punkty a otrzymasz wykres.

5) Przepisz zadanie do zeszytu.

```

Podaj liczbę x: 0.25

f(x)=0.06
jeśli powtórzyć program wcisnij klawisz a
w przeciwnym wypadku wcisnij dowolny klawisz literowy
a
Podaj liczbę x: 1

f(x)=1
jeśli powtórzyć program wcisnij klawisz a
w przeciwnym wypadku wcisnij dowolny klawisz literowy
s

Aby kontynuować, naciśnij dowolny klawisz . . .

```

## Zadanie 22

Zmienne z trzema literami nazwiska.

Napisać program, który będzie można uruchomić przez podanie kodu dwu-literowego. Komputer pyta się o pierwszą literę kodu (pierwsza litera Twojego imienia), a następnie o drugą literę kodu (pierwsza litera Twojego nazwiska) następnie, jeśli jest prawidłowy szyfr, to komputer drukuje "DZIEŃ DOBRY", jeśli jest zły "ŻEGNAJ". Przy podawaniu szyfru wielkość liter nie będzie miała znaczenia. Użyj jednej instrukcji if else. Gdy program działa, zmień instrukcję if else na wersję skróconą patrz przykład poniżej. Gdy wpisujesz treść programu do zeszytu zapisz tak, aby można było zobaczyć obie wersje(normalną i skróconą if else).

Przykład:

C++ pozwala na krótszy zapis instrukcji warunkowej:

```
(a>b)? MAX=a : MAX=b;
      inaczej:
if (a>b) MAX=a; else MAX=b;
```

```
Proszę podać pierwszą literę kodu:
A
Proszę podać drugą literę kodu:
b
Podany kod to: Ab
Dzień Dobry
```

**Zadanie 23**

Zmienne z trzema literami nazwiska.

Napisać program, taki, że Komputer będzie wykonywał program (powtarzał napis DZIEŃ DOBRY) aż do wczytania liczby z przedziału  $[-3,2)$ .

Uzupełnij tabelę w zeszycie:

Wartość wczytanej liczby	Reakcja komputera: Możliwe wpisy do tabeli:
	<ul style="list-style-type: none"> <li>Powtórzył</li> <li>Nie powtórzył</li> </ul>
-30	
-4	
-3	
0	
2	
10	
15	

```
DZIEŃ DOBRY
będę powtarzał program, aż do wczytania liczby z przedziału <-3,2)
Podaj liczbę x: 3

DZIEŃ DOBRY
będę powtarzał program, aż do wczytania liczby z przedziału <-3,2)
Podaj liczbę x: -3

jeśli powtórzyć program wcisnij klawisz t
w przeciwnym wypadku wcisnij dowolny klawisz literowy
t

DZIEŃ DOBRY
będę powtarzał program, aż do wczytania liczby z przedziału <-3,2)
Podaj liczbę x: _
```

**Przykład 16**

Temat: Program oblicza pole oraz obwód okręgu po podaniu promienia. Sprawdza wczytywane dane oraz wybierana jest opcja obliczeń.

Wykonaj:

- 1)Przepisz temat,
- 2)Uruchom przykład
- 3)Uzupełni tabelę w zeszycie:

Wielkość wczytywana	Odpowiedź programu, ( co pokazało się na ekranie).
Opcja=3	
Opcja=1	
R= - 1	
R=2	
Opcja=2	
R= - 3	
R=3	

- 4)Wpisz przykład do zeszytu
- 5)W zeszycie otocz ramką tą linijkę, która tworzy stałą w przykładzie

```
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <conio.H>
using namespace std;
int main(int argc, char *argv[])
{
    float const PI=3.14159; short int opcja;
    float pole,obwod,r;
    printf("Umiem obliczac pole oraz obwod okregu\n");
    printf("po podaniu promienia\n");
    printf("umiem sprawdzac wczytywane dane\n");
    printf("musisz wybrac opcje\n");
    printf("*****\n");
    printf("1 -pole okregu\n");
    printf("2-obwod okregu\n");
    printf("*****\n");
    do
    {
        printf("podaj opcje\n");
        scanf("%d",&opcja);
    }
    while((opcja!=1)&&(opcja!=2));
    do
    {
        printf("podaj promien okregu\n");
        scanf("%f",&r);
    }
```

```

while (r<=0);
printf("wczytane dane");
printf("\nr=%.2f",r);
printf(" cm");
printf("\nwyniki");
if (opcja==1)
{
    pole=PI*r*r;
    printf("\npole=%.2f",pole);
    printf(" cm2");
}
if (opcja==2)
{
    obwod=2*PI*r;
    printf("\nobwod=%.2f",obwod);
    printf(" cm");
}

    cout<<endl;
system("PAUSE");
return EXIT_SUCCESS;
}

```

#### **Zadanie 24**

Zmienne z trzema literami nazwiska.

Napisz program, który obliczać będzie

- 1.pole trójkąta
- 2.promień koła opisanego
- 3.promień koła wpisanego

po podaniu długości boków.

Program sprawdza poprawność wprowadzenia danych tzn. czy  $A, B, C > 0$  oraz czy boki tworzą trójkąt. {jaki to warunek? }

Komunikat o spełnieniu tego warunku podaj w kolorze zielonym, a o niespełnieniu w kolorze czerwonym.

#### **Dane do sprawdzenia:**

A=3 B=4 C=5

#### **Wyniki:**

S=6.00 cm<sup>2</sup>

R= 2.50 cm

r=1 cm

#### **Wzory:**

p– połowa obwodu trójkąta

S– pole trójkąta

r– promień okręgu wpisanego w trójkąt

R– promień koła opisanego na trójkącie

A, B, C boki trójkąta

$$p = \frac{(A + B + C)}{2}$$

$$S = \sqrt{p(p - A)(p - B)(p - C)}$$

$$r = \frac{S}{p}$$

$$R = \frac{(ABC)}{(4S)}$$

```

Umieć obliczać pole trójkąt oraz promień koła wpisanego lub opisanego na trójkąt
ie
po podaniu długości boków trójkąta
umieć sprawdzać wczytywane dane
musisz wybrać opcje
*****
1-pole trójkąta
2-promień koła opisanego na trójkącie
3-promień koła wpisanego w trójkąt
*****
podaj opcje
1
podaj długość boku A w centymetrach:
1
podaj długość boku B w centymetrach:
9
podaj długość boku C w centymetrach:
1
boki nie tworzą trójkąta
podaj długość boku A w centymetrach:
3
podaj długość boku B w centymetrach:
4
podaj długość boku C w centymetrach:
5
boki tworzą trójkąt
wczytane dane
A=3.00 cm
B=4.00 cm
C=5.00 cm
wyniki
pole trójkąta= 6.00 cm2
Aby kontynuować, naciśnij dowolny klawisz . . . _

```

### Zadanie 25

Zmienne z trzema literami nazwiska.

Napisz program, który będzie rozwiązywał równanie kwadratowe w postaci

$$A \cdot x^2 + B \cdot x + C = 0.$$

Sprawdzone będzie czy  $A \neq 0$ . Rozwiązania podawane będą w zależności od wartości delty.

<u>Dane do sprawdzenia:</u> A=1 B=1 C=1 <u>Wyniki:</u> Delta= -3 Brak rozwiązań	<u>Dane do sprawdzenia:</u> A=1 B=2 C=1 <u>Wyniki:</u> Delta= 0 Jedno rozwiązanie X0= - 1	<u>Dane do sprawdzenia:</u> A=1 B=2 C= -3 <u>Wyniki:</u> Delta = 16 Dwa rozwiązania X1= -3 X2= 1
---	--	--

```

podaj wartość współczynnika A: 0
podaj wartość współczynnika A: 1
podaj wartość współczynnika B: 2
podaj wartość współczynnika C: -3
to równanie ma dwa pierwiastki rzeczywiste: X1=-3 oraz X2=1
Delta=16
Aby kontynuować, naciśnij dowolny klawisz . . .

```

**Zadanie 26**

Zmienne z trzema literami nazwiska.

Napisz program, który będzie rozstrzygał po wczytaniu trzech różnych liczb A B C , która jest największa , średnia i najmniejsza np.

A- średnie

B- największe

C- najmniejsze

Sprawdzone będzie również czy liczby; są od siebie różne. W przypadku podania liczb o tej samej wartości to komputer wyda stosowny komunikat np. A=B lub A=B=C itp. i poprosi o ponowne wczytanie liczb. Stosowny komunikat wypisz kolorem czerwonym.

```

podaj wartość liczby A: 45
podaj wartość liczby B: 45
podaj wartość liczby C: 23
liczby o tej samej wartości podaj ponownie wartości liczb
podaj wartość liczby A: 45
podaj wartość liczby B: 67
podaj wartość liczby C: 23
Największa=67
Średnia=45
Najmniejsza=23
Aby kontynuować, naciśnij dowolny klawisz . . .

```

**INSTRUKCJE break i continue.**

Instrukcja **break** powoduje natychmiastowe bezwarunkowe opuszczenie pętli dowolnego typu i przejście do najbliższej instrukcji po zakończeniu pętli. Jeśli w pętli for opuścimy wyrażenie logiczne, to zostanie automatycznie przyjęte 1. Pętla będzie, zatem wykonywana bezwarunkowo w nieskończoność

Instrukcja **continue** powoduje przedwczesne, bezwarunkowe zakończenie wykonania wewnętrznej instrukcji pętli i podjęcie próby realizacji następnego cyklu pętli. Próby, ponieważ najpierw zostanie sprawdzony warunek kontynuacji pętli.

**INSTRUKCJE switch i case.**

Instrukcja switch dokonuje WYBORU w zależności od stanu wyrażenia przełączającego (selector) jednego z możliwych przypadków - wariantów (case). Każdy wariant jest oznaczony przy pomocy stałej - tzw. ETYKIETY WYBORU. Wyrażenie przełączające może przyjmować wartości typu int. Ogólna postać instrukcji jest następująca:

```

switch (selector)
{
    case STAŁA1: Ciąg_instrukcji-wariant 1;
    case STAŁA2: Ciąg_instrukcji-wariant 2;
    .....
    case STAŁAn: Ciąg_instrukcji-wariant n;
    default : Ostatni_ciąg_instrukcji;
}

```



Należy podkreślić, że po dokonaniu wyboru i skoku do etykiety wykonane zostaną również WSZYSTKIE INSTRUKCJE PONIŻEJ DANEJ ETYKIETY. Jeśli chcemy tego uniknąć, musimy dodać rozkaz break.

Innymi słowy, wyrażenie warunkowe switch, jest mówiąc krótko, bardzo rozbudowaną postacią instrukcji if. Wyrażenia switch można by w ogóle nie używać, ponieważ zastąpić je można całkowicie seriami wyrażenia if...else, jednak switch ma jedną wielką przewagę – **czytelność**. Konstrukcję if...else czyta się „jeżeli warunek jest spełniony, zrób to, a przeciwnym przypadku...”, podczas gdy wyrażenie switch odpowiada raczej sformułowaniu „w zależności od wartości zrób to, to, to lub to”.

Seria instrukcji case umieszczona w nawiasach klamrowych obsługuje konkretne wartości tej zmiennej. Nie musimy obsługiwać wszystkich. Jeżeli specjalnej obsługi wymaga tylko kilka z nich, a pozostałe mogą być obsługiwane we wspólny sposób lub wręcz zignorowane, na końcu wyliczenia, po ostatnim wystąpieniu należy umieścić instrukcję **default** i za nią, po dwukropku podajemy instrukcję, które mają być wykonywane (lub nic nie podajemy, wtedy wszystkie przypadki poza tymi obsługiwany przez case zostaną zignorowane).

### #define.

Użycie #define polega na zastąpieniu w tekście programu jednych łańcuchów znaków przez inne.

np.

```
# define Program main()
# define Begin {
# define Writeln printf
# define Readln getch()
# define End }
```

Takie zdefiniowanie pozwala zastąpić instrukcje CPP instrukcjami Pascala.

### Przykład 17

Zapisz w zeszycie składnię instrukcji

- INSTRUKCJE switch i case;
- znaczenie break oraz default;
- #define.

### Treść programu

Program po podaniu numeru dnia tygodnia odpowie, jaki to dzień tygodnia. W przykładzie zastosowano #define.

Wykonaj:

- Wpisz przykład do komputera oraz przetestuj go.
- Wykonaj schemat blokowy przykładu.

```
#define pisz printf //zastąpieniu w tekście programu jednych łańcuchów znaków printf przez
pisz.
#include <cstdlib>
#include <iostream>
#include <stdio.h>
using namespace std;
```

```

int main(int argc, char *argv[])
{
    int Numer_Dnia;
    pisz("\nPodaj numer dnia tygodnia\n");
    scanf("%d",&Numer_Dnia);
    switch(Numer_Dnia)
    {
        case 1: {pisz("PONIEDZIALEK");break;}
        case 2: {pisz("WTOREK");break;}
        case 3: {pisz("SRODA");break;}
        case 4: {pisz("CZWARTEK");break;}
        case 5: {pisz("PIATEK");break;}
        case 6: {pisz("SOBOTA");break;}
        case 7: {pisz("NIEDZIELA");break;}
        default: pisz("\nniema takiego dnia tygodnia!!!");
    }
    cout<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

Zwróć uwagę, że w przykładzie wariant default zostanie wykonany ZAWSZE nawet, jeśli podasz liczbę większą niż 7.

### Zadanie 27

Temat: Użycie instrukcji case, swich, oraz #define

Wykonaj:

a) Program z przykładu poprzedniego zmodyfikuj w taki sposób, aby: po podaniu pierwszej litery kraju, z którym graniczy Polska odpowie, jaki to kraj.  
Np. po podaniu „c” lub „C” wyświetlone zostanie „Czechy”. Wielkość liter nie będzie miała znaczenia.

Zmienne z trzema literami nazwiska np. kraj\_kow gdy uczeń nazywa się Kowalski.

b) W przykładzie zastosowano #define do przeddefiniowania Ty zastosuj:

**case** → na **przypadek**

**switch** → na **przełącz**

c) Zmienną selektora zadeklaruj jako char.

Zmień literkę d na taką aby wczytywać znaki(stringi)

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

Do sprawdzania użyj znaku apostrofu np. ‘C’

d) Zmienne z trzema literami nazwiska np. kraj\_kow gdy uczeń nazywa się Kowalski.

Koniec zadania 27

```

Podaj pierwszą literę kraju, z którym graniczy Polska:
n
Niemcy
Aby kontynuować, naciśnij dowolny klawisz . . .

```

### Skok bezwarunkowy goto (idź do...),

Gdy chcesz, aby program przeszedł do określonego miejsca w programie możesz użyć tak zwanej **etykiety**.

Etykieta składa się z:

- polecenia **goto** nazwa; np. goto skocz;
- oraz nazwy zakończonej dwukropkiem np. **dawaj:** gdzie trzeba skoczyć.

UWAGA: Po każdej etykiecie musi wystąpić co najmniej jedna instrukcja. Jeśli etykieta oznacza koniec programu, to musi po niej wystąpić instrukcja pusta. Należy tu zaznaczyć, że etykieta nie wymaga deklaracji.

np.

goto dawaj;

// część programu

dawaj:

### Przykład 18

Temat: Po wczytaniu liczby program poda jaką literę mamy w kodzie ASCII o liczbę miejsc dalej od litery A.

Wykonaj:

- Zapisz w zeszycie co to jest etykieta, jak jest zbudowana, jakie jest znaczenie znaku „:”, przepisuj również uwagę.
- odczytaj z tabeli kodów ASCII numer kodu dla litery A oraz K o ile numerów się różnią
- Wpisz przykład i uruchom.
- W zeszycie zanotuj, co oznacza:
  - `printf("%c", 'A');`
  - `printf("\t\t\t\t\t%d", 'A');`
  - `printf("\n%c\t\t\t\t\t%d\n", 'A'+liczba, 'A'+liczba);`

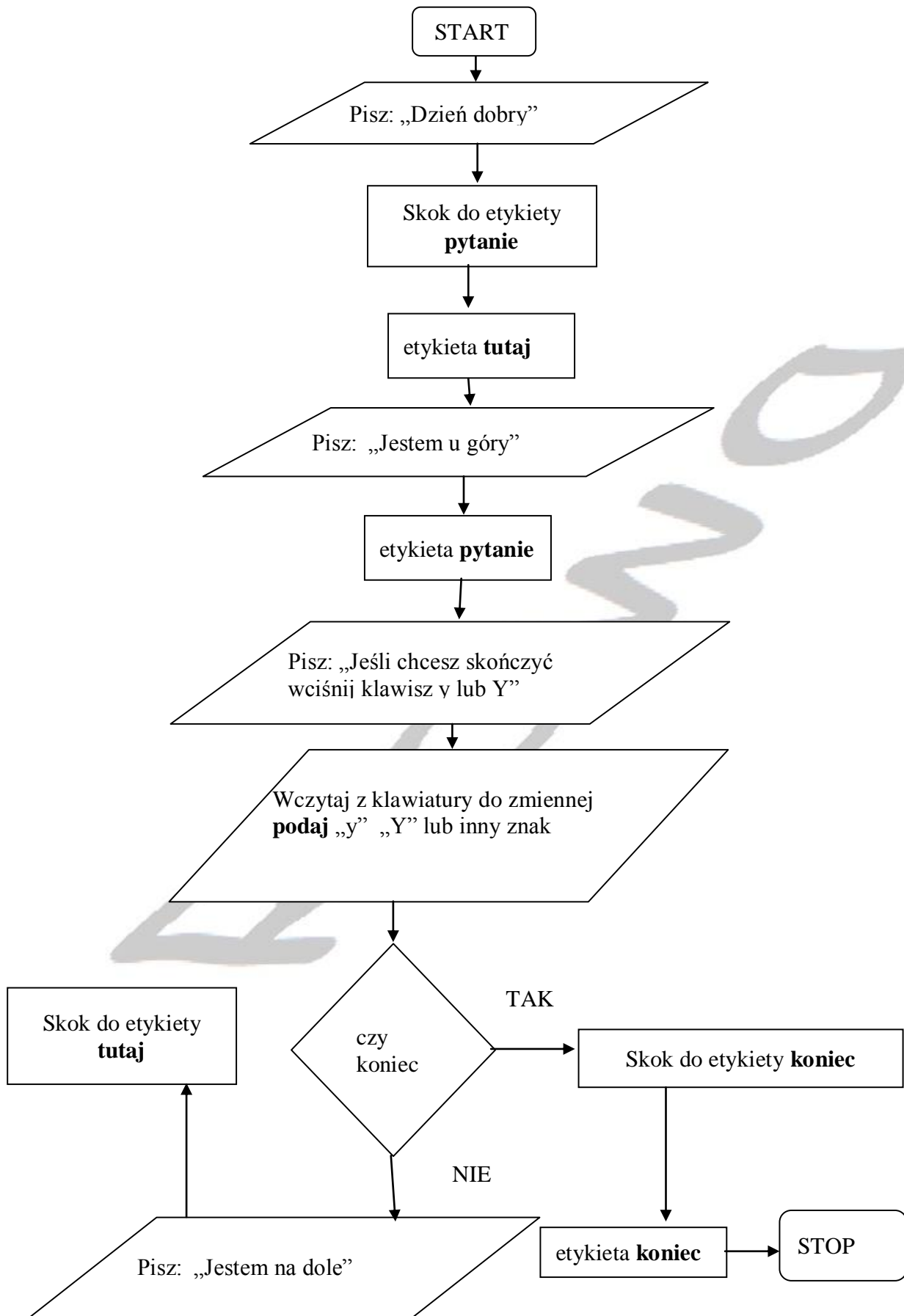
Przepisz linijkę programu i dokładnie opisz znaczenie zastosowanych wszystkich znaków.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    short int liczba;
    printf("Wydrukuj A jako \nLiteral znakowy:\tKod ASCII:\n");
    printf("%c", 'A');
    printf("\t\t\t\t\t%d", 'A');
    skocz:                                // miejsce skoku
    printf("\nnumiem podac kod ASCII o n wiekszy od kodu A ");
```





```

Dzień Dobry

Jeśli chcesz skończyć wciśnij klawisz y lub Y
Podaj jakiś znak: n
Jestem na dole
Jestem u góry
Jeśli chcesz skończyć wciśnij klawisz y lub Y
Podaj jakiś znak: y
Aby kontynuować, naciśnij dowolny klawisz . . .

```

### Pamiętaj!!!

Jeśli stosujesz scanf dla zmiennej typu char to wczytywane są dwa znaki jeden to kod ENTER (stąd podwójna pętla, gdzie w pierwszej zamiast znaku wczytywany jest enter). W tym wypadku zamień scanf na cin.

### Różnica między instrukcją getch() a kbhit()

#### a) kbhit()

Aż do naciśnięcia dowolnego klawisza.

#### b) !kbhit()

Zapis !kbhit() oznacza "nie naciśnięto klawisza", czyli w buforze klawiatury nie oczekuje znak.

#### c) getch()

Zwróć uwagę, że funkcja getch() może oczekiwać na klawisz w nieskończoność. Aby uniknąć kłopotliwych sytuacji, czasem znacznie wygodniej jest zastosować kbhit(), szczególnie, jeśli czekamy na dowolny klawisz.

### Zadanie 29

Babcia Aurelia dorabia sobie do skromnej emeryturki sprzedając na rynku owoce z własnego sadu. Do tego celu używa wagi szalkowej. Zasada działania takiej wagi opiera się na dostawianiu ciężarków bądź to na szalce lewej bądź na prawej, aż do uzyskania momentu zrównoważenia szalek. Niestety zdarzył się wypadek. Babci ktoś „podprowadził” ciężarki. No i babcia popadła w kłopot. Nie mając ciężarków nie mogła sprzedawać na rynku. Hmm...

Ale, Aurelia to zaradna babka, zeszła do piwnicy, a tam wygrzebała duży pręt. Zabrała go do Staśka sprytnego ślusarza. Ten zważył pręt, wyszło 40kilogramów, a następnie podzielił go na 4 części, tworząc babci odważniki w taki sposób że, można było zważyć na wadze szalkowej każdy ciężar do wagi 40kg. Podaj jakie wagi miały poszczególne części pręta.

Program powinien dokonać sprawdzenia wprowadzanych danych pod kątem:

- czy suma wprowadzanych części pręta daje 40 kg?

Suma części musi dać 40 kg

Podaj wagę pierwszej części:

- czy nie przekroczono już liczby dostępnych ciężarków?

stop - użyłeś już 4 ciężarów - podczas jednego ważenia, możesz użyć tylko 4 ciężarków, każdy po jednym razie!

- czy nie próbowano dołożyć ciężarków o tym samym symbolu?

Dany odważnik może być użyty tylko raz podczas jednego ważenia!  
waga szalki lewej= 1

- czy ważony towar nie przekracza 40 kg?

Ciężar jaki chcemy zważyć to: 45  
 Za duży ciężar!  
 Ciężar jaki chcemy zważyć to: \_

Nadaj każdej części pręta odpowiedni symbol! (proponuję: A,B,C,D)

W trakcie pracy możesz używać instrukcji Sleep dla zrównoważenia tempa pojawiania się kolejnych instrukcji – patrz zadanie 13

Podczas wyświetlania wyniku podaj graficzny obraz ciężarków w kolorach oraz sygnał dźwiękowy podczas wyświetlania tych obrazów.

Proponuję poniższą listę kroków:

1. Wprowadź podział 40 kilowego pręta na 4 części nadając im odpowiednie symbole;
2. Dokonaj sprawdzenia czy suma wprowadzanych części pręta daje 40 kg;
3. Wyświetl wprowadzone dane na ekran;
4. Podaj komunikat: Towar ważymy na szalce prawej! ( w kolorze czerwonym!);

**Uwaga!**  
**Towar ważymy na szalce prawej!**

5. Wprowadź ciężar jak chcesz zważyć;
6. Dokładaj odważniki do odpowiednich szalek, wprowadzając symbole odważników;
7. Sprawdzaj czy nie dokładasz ciężarów o tym samym symbolu;
8. Sprawdzaj czy użyłeś już 4 ciężarów - podczas jednego ważenia, możesz użyć tylko 4 ciężarków, każdy po jednym razie!;
9. Sprawdzaj czy ważony towar nie przekracza 40 kg;
10. Podaj komunikat:
  - BRAWO - udana próba ważenia!!!!
  - Niestety - nie udało Ci się dobrać ciężarków!
11. Na oddzielnym ekranie wyświetl graficzny obraz efektu ważenia, poniżej proponowane przedstawienie graficzne.

**BRAWO - udana próba ważenia!!!!**





Dokonaj sprawdzenia czy Twoimi ciężarkami potrafisz zważyć towary o następujących wagach:

- 5kg;
- 8kg;
- 17kg;
- 22kg;
- 24kg;
- 31kg;
- 36kg;

### **Zadanie przykładowe na sprawdzian czas 85 minut**

Po wykonaniu każdego zadania zgłaszać nauczycielowi. Kolejność dowolna.

ocena: punkty      ocena

<b>0–1</b> ndst	<b>2</b>	<b>3</b>	<b>4</b> dobry	<b>5</b> bardzo dobry	<b>6</b> celujący
	dopuszczający	dostateczny			

### **Zadanie1** (jeden punkt)

Program będzie uruchamiany przez podanie kolejno : numeru dnia urodzenia+10, pierwszej litery imienia oraz pierwszej litery nazwiska. Wielkość liter nie będzie miała znaczenia

– Gdy podany jest prawidłowo szyfr komputer napisze "Brawo znasz moje parametry"

– Gdy szyfr jest nieprawidłowy komputer napisze "Nie znasz szyfru żegnaj".

```

Prosze podać pierwszą literę kodu:
37
Prosze podać drugą literę kodu:
a
Prosze podać trzecią literę kodu:
R
Podany kod to: 37aR
Brawo znasz moje parametry

```

### **Zadanie2** (jeden punkt)

Komputer napisze: będę powtarzał program czyli napis gdy podasz liczbę z przedziału (-1,3> następnie zapyta: czy chcesz powtarzać część programu piszącą "SPRAWDZIAN ZADANIE2".

Jeśli podasz liczbę z przedziału (-1,3> nastąpi wczytanie liczby i powtórzenie napisu "SPRAWDZIAN ZADANIE2", jeśli podasz liczbę z poza przedziału nastąpi wyjście z programu.

```

SPRAWDZIAN ZADANIE2
będę powtarzał program czyli napis gdy podasz liczbę z przedziału (-1,3>
Podaj liczbę x: 3

SPRAWDZIAN ZADANIE2
będę powtarzał program czyli napis gdy podasz liczbę z przedziału (-1,3>
Podaj liczbę x: 4

Aby kontynuować, naciśnij dowolny klawisz . . . _

```

**Zadanie3** (jeden punkt)

Zostanie wyświetlone menu:

- 1– obliczanie objętości walca
- 2– obliczanie powierzchni całkowitej walca

Wybór wariantu 1 lub 2 wykonanie obliczeń oraz wyprowadzenie wyników na monitor z podaniem jednostek czyli  $\text{cm}^2$  i  $\text{cm}^3$ .

Sprawdzanie poprawności wczytanych danych:

- ♠ numeru wariantu (powtórzenie MENU → gdy numer jest nieprawidłowy)
- ♠ danych liczbowych

```

Umiem obliczac objętość oraz pole całkowite walca
po podaniu wysokości oraz promienia podstawy walca
umiem sprawdzac wczytywane dane
musisz wybrac opcje
*****
1-objętość walca
2-powierzchnia całkowita walca
*****
podaj opcje
1
podaj wysokość walca:
8
podaj promień podstawy walca:
5
wczytane dane
h=8.00 cm
r=5.00 cm
wyniki
objętość walca= 628.32 cm^3
Aby kontynuować, naciśnij dowolny klawisz . . . _

```

Dla  $h=8\text{cm}$  i  $r=5\text{cm}$   $P=408,41\text{cm}^3$

**Zadanie4** (jeden punkt)

Po wczytaniu wartości dwóch kątów program poda wartość trzeciego kąta i odpowie jaki to jest trójkąt (prostokątny, równoboczny, równoramienny, różnokątny).

Zapewnij powtarzalność programu na literę „j” lub „J”.

Czyli jeśli chcesz powtórzyć podaj literę „j” lub „J”.

```

Podaj pierwszy kąt: 45
Podaj drugi kąt: 45
kąt gamma= 90
To jest trójkąt prostokątny
jeśli powtórzyć program wciśnij J lub j
J
Podaj pierwszy kąt: 45
Podaj drugi kąt: 60
kąt gamma= 75
To jest trójkąt różnokątny
jeśli powtórzyć program wciśnij J lub j

```

Sprawdzenie

alfa	beta	Komunikat o trójkącie:
60	60	Równoboczny
30	120	Równoramienny
56	90	Prostokątny
34	78	Różnokątny

**Zadanie5** (jeden punkt)

Napisz program rozpoznający, jaką cyfrę wczytał użytkownik. Użyj instrukcji switch case. Cyfra wczytywana, jako liczba naturalna. Gdy wczytana jest nieistniejąca cyfra to komputera wyda komunikat „Taka cyfra nie istnieje”. Gdy wczytana jest istniejąca cyfra to komputera poinformuje „Taka cyfra istnieje i jest nią cyfra np. zero”.

Zapewnij powtarzalność programu. Do powtarzalności użyj instrukcję goto. Zmień instrukcje gotoxy na skocz przy użyciu #define.

```
Podaj liczbę naturalną:
2
Taka cyfra istnieje i jest nią cyfra 2
jeśli zakończyć program wciśnij t/T
w

Podaj liczbę naturalną:
4
Taka cyfra istnieje i jest nią cyfra 4
jeśli zakończyć program wciśnij t/T
t

Aby kontynuować, naciśnij dowolny klawisz . . .
```

**Zadanie6** (jeden punkt)

Z użyciem pętli „do while” lub „while” napisz program wykonujący rozkład liczby na czynniki pierwsze. Podajemy liczbę naturalną i ukazują się rozkład w formie jak poniżej:

Dla liczby N=8778

Na ekranie uzyskasz

8778=2\*3\*7\*11\*19

```
Podaj liczbę: 8778
Czynniki pierwsze liczby 8778 : 2 3 7 11 19
Aby kontynuować, naciśnij dowolny klawisz . . . _
```