

Programowanie w języku C++ (dotycząca Dev-C++)

Część 1

Edytor, Kompilator.

Tryb tekstowy



Dev-C++

System DEV-C++ składa się z:

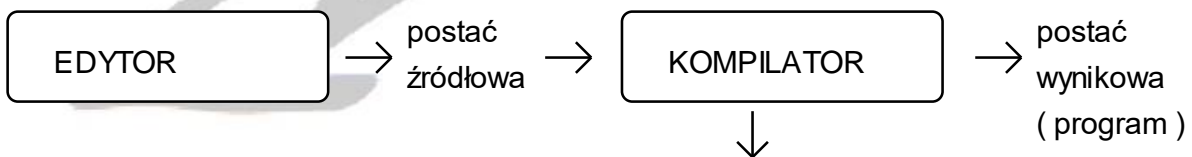
1. **EDYTORA** – służy do tworzenia tekstu programu w języku C++, traktowanego jako ciąg znaków (litery+liczby+znaki specjalne). Tekst taki nazywamy programem w postaci źródłowej.

2. **KOMPILATORA** – program tłumaczący program w postaci źródłowej na program w postaci kodu asemblera (jest to ciąg operacji maszynowych zakodowanych w postaci liczb dwójkowych). Otrzymujemy plik pośredni z rozszerzeniem **OBJ**.

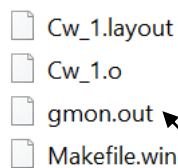
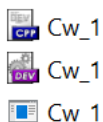


Wiesz już, że **kompilacja** to tłumaczenie napisanego przez Ciebie tekstu zawierającego instrukcje języka C++ (czyli kodu źródłowego) na formę zrozumiałą dla komputera (czyli na język maszynowy). O dziwo jednak, efektem kompilacji nie jest jeszcze gotowy do uruchomienia plik, czyli tzw. **plik wykonywalny**. Dlaczego? Ponieważ prawie każdy program pisany w C++ korzysta z bibliotek, zwykle również kod źródłowy jest dla wygody dzielony na wiele plików, (a każdy z takich plików, oraz biblioteki są osobno kompilowane), musi jeszcze zajść proces, który połączy pliki źródłowe oraz biblioteki statyczne w jeden plik wykonywalny, a także ustali relacje tego pliku z bibliotekami dynamicznymi. Ten proces nazwa się **linkowaniem** czyli łączeniem (ang. *to link* - łączyć), a program odpowiedzialny za ten proces - **linkerem**, albo bardziej po polsku - **konsolidatorem**. Dopiero po tym etapie mamy już gotowy prawdziwy, działający (jeśli wszystko poszło dobrze!) program.

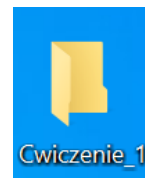
Użycie **linkera (konsolidatora)** powoduje zatem połączenie plików *.obj, *.lib, *.dll, oraz wygenerowanie pliku wykonywalnego *.EXE. Gdy program źródłowy ma błędy kompilator sygnalizuje błąd poprzez wskazanie jego miejsca oraz podanie jego numeru.



Ćwiczenie 1.(wykonaj praktycznie)



Utwórz nowy katalog o nazwie Cwiczenia. W nim umieścisz katalogi do kolejno wykonanych sześciu ćwiczeń praktycznych. Dla pierwszego ćwiczenia utwórz katalog



Tworzenie pierwszego projektu dla trybu DOS.

Plik→Nowy→Projekt→Console Application→Ok

Pojawi się okno gdzie będziemy przechowywać nasz projekt→wybierz miejsce przechowywania i zapisz w katalogu Cwiczenie_1 .

Tak powinna wyglądać zawartość katalogu, po prawidłowo wykonanych

operacjach.

W okienku edycji kodu źródłowego automatycznie jest **generowany szkielet** aplikacji:

```
#include <iostream>

/* run this program using the console pauser or add your own getch, system("pause") or input loop */

int main(int argc, char** argv)
{
    return 0;
}
```

Aby użyć przestrzeni nazw std należy przed funkcją główną dopisać komendę:

using namespace std; - użyj przestrzeni nazw standard; (poprzednie wersje dev-a generowały tę komendę automatycznie)

Za pomocą słów kluczowych **using namespace** informujemy kompilator, że chcemy aby wszystkie funkcje, klasy i szablony należące do przestrzeni nazw nie wymagały przedrostka.

Czyli, polecenie to rozwiązuje problem dublowania się nazw różnych funkcji i poleceń.

Std oznacza bibliotekę standardową, w której znajdują się definicje wszystkich najważniejszych symboli oraz poleceń i funkcji.

Kompilowanie programu czyli tłumaczenie z języka CPP na język maszynowy wraz z wykrywaniem błędów. Uruchom → Kompiluj lub Ctrl+F9

Uruchomienie programu gdy kompilacja nie miała błędów to Uruchom → Uruchom lub Ctrl+F10

Pliki generowane przez kompilator:

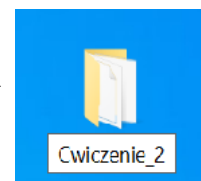
- plik projektu *.dev
- plik kodu źródłowego *.cpp
- plik wykonywalny *.exe
- plik binarny *.o
- plik reguł kompilatora Makefile. win

Ćwiczenie.2.

Zapisu dokonaj w nowo stworzonym katalogu o nazwie cwiczenie_2

Polskie znaki na konsoli:

- W treści programu wpisz instrukcję system("chcp 1250");
- uruchom Właściwości okna CMD (poprzez kliknięcie szarego tytułu okna) i w zakładce Czcionki należy wybrać czcionkę Lucida Console i zatwierdzić dolną opcją.



Ćwiczenia do wykonania pisemnego (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

Narysuj oraz opisz schemat współpracy edytora z kompilatora.

Pytanie 2

Co to jest postać źródłowa programu.

Pytanie 3

Co to jest kompilator? Jakiego wykonuje czynności.

Pytanie 4

Co to jest kod asemblera?

Pytanie 5

Jakie pliki mają rozszerzenia OBJ.?

Pytanie 6

Jakie czynności wykonuje linker(konsolidator)?

Pytanie 7

Jakie rozszerzenie ma plik wykonywalny?

Pytanie 8

b) Zapisz w jaki sposób tworzymy nowy projekt dla trybu DOS.

Pytanie 9

c) Zapisz szkielet aplikacji automatycznie generowanej przez system.

Pytanie 10

Wypisz oraz opisz pliki generowane przez system CPP Dev.

Pytanie 11

Zapisz w jaki sposób kompilujemy programy w DEV

Pytanie 12

Zapisz w jaki sposób uruchamiamy programy w DEV

Pytanie 13

Co to jest komentarz, rodzaje komentarzy z przykładem, jakim kolorem zaznaczany jest komentarz w Dev.

Pytanie 14

Co to są pliki nagłówkowe. Po jakim słowie kluczowym są pisane pliki nagłówkowe i jak się kończy deklaracja. Znaczenie znaku #. Jakim kolorem oznaczane są dyrektywy preprocesora.

Pytanie 15

Co to jest funkcja główna. Przepisz przykład. Co oznaczają puste nawiasy okrągłe? Co oznaczają nawiasy klamrowe? Znaczenie int.

Pytanie 16

Opisz instrukcję cout

Pytanie 17

w jaki sposób realizujemy przejście do nowej linii z użyciem instrukcji cout

Pytanie 18

omów użycie znaków: # { } ;

Pytanie 19

Opisz polecenie using namespace std;

Pytanie 20

Opisz w jaki sposób używamy instrukcji dosowych. Zapisz cztery przykłady takich funkcji.

Pytanie 21

zapisz uwagi na temat stosowania dużych i małych liter.

Pytanie 22

zapisz uwagi na temat stylu programowania.

Pytanie 23 → pytanie praktyczne.

Na domowym komputerze zainstaluj C++ DEV możliwie najnowszą wersję. Uruchom program. Wstaw szkielet aplikacji automatycznie generowanej przez system. Wykonaj rzut ekranu z wstawionym i widocznym szkieletem aplikacji z Twoim Nazwiskiem. Zapisz rzut

ekranu na pendrive oraz wyślij na pocztę w formacie GIF lub JPG. Zrzut ten będzie sprawdzany łącznie z pytaniami z pracy domowej.

Ćwiczenie .3.

Komentarz

Komentarze są to napisy, których kompilator nie bierze pod uwagę podczas kompilacji programu. W programie zademonstruj działanie obu typów komentarzy:

a) Komentarz wielolinijkowy.

`/* treść komentarza */`

b) Komentarz jednoliniowy

Komentarz zaczynający się od `//` treści komentarza.

Wstaw dowolne napisy jako komentarze.

Ćwiczenie zapisz w katalogu o nazwie ćwiczenie_3.

Plik nagłówkowy

Każdy program w C++ musi najpierw załączać odpowiednie pliki nagłówkowe z definicjami interesujących nas funkcji (można pisać własne pliki nagłówkowe). Po tych liniijkach ze słowem **#include** (możemy załączać kilka plików, ale wtedy każdemu poświęcamy oddzielną instrukcję `#include`). Zauważcie, że po instrukcji `#include` nie ma średnika. Jest ich kilkadziesiąt, a w każdym jest zdefiniowane ok. 50 funkcji.

Opis plików nagłówkowych..

CONIO.H - CONsole Input/Output.

Plik nagłówkowy zawierający prototypy funkcji potrzebnych do obsługi standardowego Wejścia/Wyjścia na/z konsoli (CONsole). Plik zawiera między innymi prototyp funkcji `clrscr()`, potrzebnej nam do czyszczenia ekranu.

STDIO.H - STanDard Input/Output

Plik nagłówkowy zawierający prototypy funkcji potrzebnych do obsługi standardowego Wejścia/Wyjścia na/z konsoli (Input - Wejście, Output - Wyjście). Plik zawiera między innymi prototyp funkcji **printf()**, potrzebnej nam do drukowania wyników na ekranie.

IOSTREAM

Dołącza plik nagłówkowy udostępniający operacje we/wy w stylu C++

Ćwiczenie .4.

Instrukcja **cout** wyprowadza dane z użyciem `<<` na ekran:

1. wartości zmiennych, np. `cout<<x1;`
2. tekstów np. `cout<<"Hello Word"` (pamiętaj o użyciu cudzysłowów " ")
3. może być też do pliku.
4. `cout<<endl;` oznacza przejście do nowej linii i może być używane do wykonania pustej linii.

W tym ćwiczeniu użyj jako wzoru, przykładowego widok okna projektu:

```

#include <cstdlib>
#include <iostream>

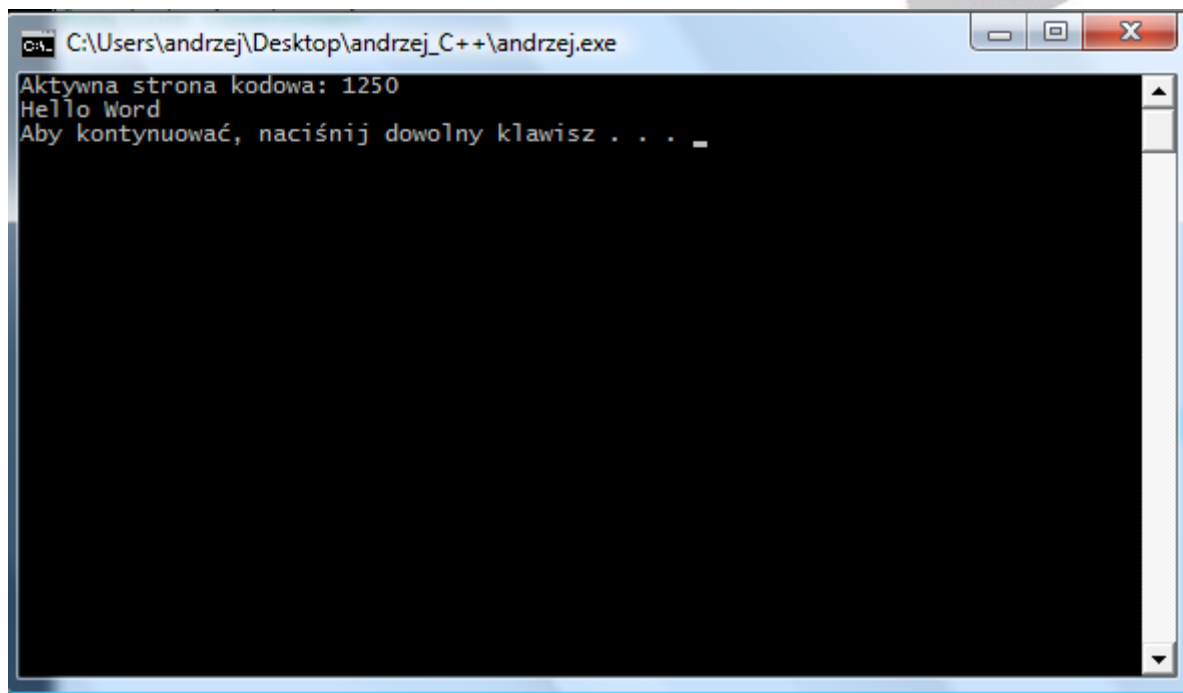
using namespace std;

int main(int argc, char *argv[])
{
    system("chcp 1250");//Tutaj wstawiam kodowanie polskich znaków
    cout<<"Hello Word";
    cout<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}

```

Ćwiczenie zapisz w katalogu o nazwie ćwiczenie_4.

Uruchomione okno CMD wygląda tak:



```

C:\Users\andrzej\Desktop\andrzej_C++\andrzej.exe
Aktywna strona kodowa: 1250
Hello Word
Aby kontynuować, naciśnij dowolny klawisz . . . _

```

Funkcja główna

Każdy program w C++ musi posiadać funkcję o nazwie **main**. Po słowie main występują nawiasy okrągłe. Puste nawiasy oznaczają, że funkcja nie przyjmuje żadnych argumentów. A sama funkcja nazywa się main (z angielskiego: główna), ponieważ to właśnie instrukcje umieszczone wewnątrz niej (w jej ciele) będą wykonywane przez program. Początek i koniec określania ciała funkcji oznaczamy odpowiednio: otworzonym i zamkniętym nawiasem klamrowym.

int oznacza typ wyniku zwracanego przez funkcję główną w tym przypadku jest to liczba całkowita.

Dev generuje automatycznie funkcję główną w postaci `int main(int argc, char *argv[])`. W nawiasie okrągłym zawarte są argumenty wejściowe funkcji.

```
np.
int main( )
{
    treść programu
}
```

Znaki specjalne

znak

występuje przed dyrektywami preprocesora.

średnik ;

stawiany jest na zakończenie polecenia(instrukcji)

nawiasy klamrowe { }

umożliwiają tworzenie bloku funkcji, czyli łączyć wiele instrukcji prostych tak, aby były wykonywane razem.

Użycie instrukcji dosowych

a)dołącz

```
#include <iostream>
```

```
#include <cstdlib>
```

b)wpisz instrukcję →pamiętaj, że instrukcja dosowa jest w cudzysłowu

```
system("dir");  lub
```

```
system("cls"); →czyszczenie ekranu
```

```
system("pause");→zatrzymanie programu
```

która zatrzyma program i wyświetli komunikat mniej więcej w tym stylu: **Aby kontynuować.**

naciśnij dowolny klawisz...

```
system("color 5");→ustawienie koloru tła
```

Ustawia domyślne kolory tła i pierwszego planu.

COLOR [atr]

atr Określa atrybut koloru dla wyjścia konsoli

Atrybuty kolorów są określone przez DWIE cyfry heksadecymalne

-- pierwsza oznacza tło, --druga pierwszy plan.

Każda cyfra może być jedną z wartości:

0 = Czarny

1 = Niebieski

2 = Zielony

3 = Błękitny

4 = Czerwony

5 = Purpurowy

6 = żółty

7 = Biały

8 = Szary

9 = Jasnoniebieski

A = Jasnozielony

B = Jasnobłękitny

C = Jasnoczerwony

D = Jasnopurpurowy

E = Jasnożółty

F = Jaskrawobiały

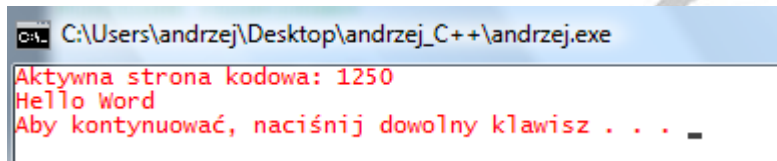
Jeśli nie podano argumentu, używany jest kolor odpowiadający chwili uruchomienia CMD.EXE. Wartość ta jest brana z bieżącego okna konsoli, z opcji /T wiersza polecenia lub z wartości rejestru DefaultColor.

Polecenie COLOR ustawia ERRORLEVEL na 1, jeśli podjęto próbę określenia tej samej wartości dla tła i dla pierwszego planu w poleceniu COLOR.

Ćwiczenie.5.

"COLOR fc" daje kolor jasnoczerwony na jaskrawobiałym tle. Zmień kolorystkę przykładu z ćwiczenia 4 wg, poniższego przykładu.

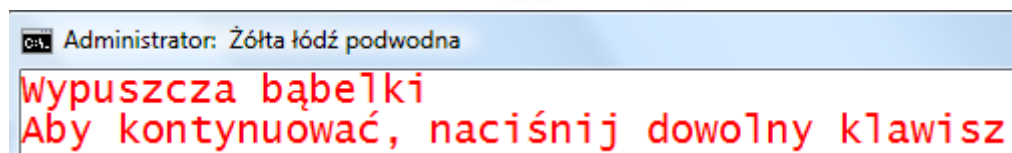
Ćwiczenie zapisz w katalogu o nazwie ćwiczenie_5.



```
C:\Users\andrzej\Desktop\andrzej_C++\andrzej.exe
Aktywna strona kodowa: 1250
Hello Word
Aby kontynuować, naciśnij dowolny klawisz . . . -
```

Ćwiczenie .6.

Używając komend systemowych, w tym komendy title, zmodyfikuj ćwiczenie 5, tak aby otrzymać następujący widok konsoli: Ćwiczenie zapisz w katalogu o nazwie ćwiczenie_6.



```
Administrator: Żółta łódź podwodna
wypuszcza bąbelki
Aby kontynuować, naciśnij dowolny klawisz
```

Uwagi:

- Pamiętaj, że dla języka C++:
PRINTF i printf to nie to samo! Słowa kluczowe i nazwy standardowych funkcji **muszą być pisane małymi literami !!!**

Styl programowania

- Można funkcję main zapisać w takiej postaci, jak to zrobiliśmy w naszym programie (czyli w czterech liniach: nazwa main, nawias, jedna pusta linijka i nawias). Ale można też to wszystko zapisać w jednej linijce, np. main(){ }. Taki sposób nie jest zalecany ze względu na czytelność programu.
- Poza nielicznymi wyjątkami (nie można przedzielać tekstów ujętych w cudzysłów) w języku C++ możemy zrobić przerwę gdzie tylko chcemy.
- void - pusty, wolny

Strumienie

Strumień to przepływ informacji od jednego urządzenia do innego. Strumieniem może być przepływ danych np. tekstu lub wartości zmiennych z pamięci komputera na ekran monitora. Trójkątne nawiasy (<< lub >>) wskazują kierunek przepływu informacji.

Obiekty

Obiekt podobnie jak program komputerowy jest to grupa danych i funkcji działających wspólnie i przeznaczonych razem do wykonania jakichś zadań. Dla przykładu obiekt **cout** służy do obsługi przesyłania danych na ekran monitora.

Przykład 1**Temat:**

Przykład demonstruj:

- dołączanie plików nagłówkowych,
- wyprowadzanie tekstów na ekran,
- użycie komentarz,
- wprowadzenie pustej linii ekranu,
- zatrzymanie programu.

Wykonaj:

- Wpisz temat do zeszytu
- Wpisać Przykła1 (łącznie z komentarzem) do komputera. Nazwa pliku na dysku p1_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.

```
//Początek przykładu1
/* Oto przykład program pokazujący wyprowadzanie tekstu na ekranu sposób1 z użyciem
obiekту cout oraz zatrzymanie działania programu */

#include <iostream>
#include <stdlib.h>
using namespace std;

int main(int argc, char *argv[])
{
    cout<<"dzień dobry, tutaj Twój komputer";
    cout<<endl;
    cout<<endl;
    return 0;
}
```

czy tekst "dzień dobry, tutaj Twój komputer" wyświetla się z polskimi znakami jeśli nie, wstaw kodowanie polskich znaków tak jak to robiłeś wcześniej np. system("chcp 1250"); pamiętaj aby sprawdzić czy podczas uruchamiania Właściwości okna CMD (poprzez kliknięcie granatowego tytułu okna) i w zakładce Czcionki wybrałeś czcionkę Lucida Console i zatwierdziłeś dolną opcją.

Przykład 2**Temat:**

Przykład demonstruj:

- wyprowadzanie tekstów na ekran z użyciem instrukcji Printf,

Wykonaj:

- Wpisz temat do zeszytu
- Wpisać Przykła2 do komputera. Nazwa pliku na dysku p2_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.


```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;

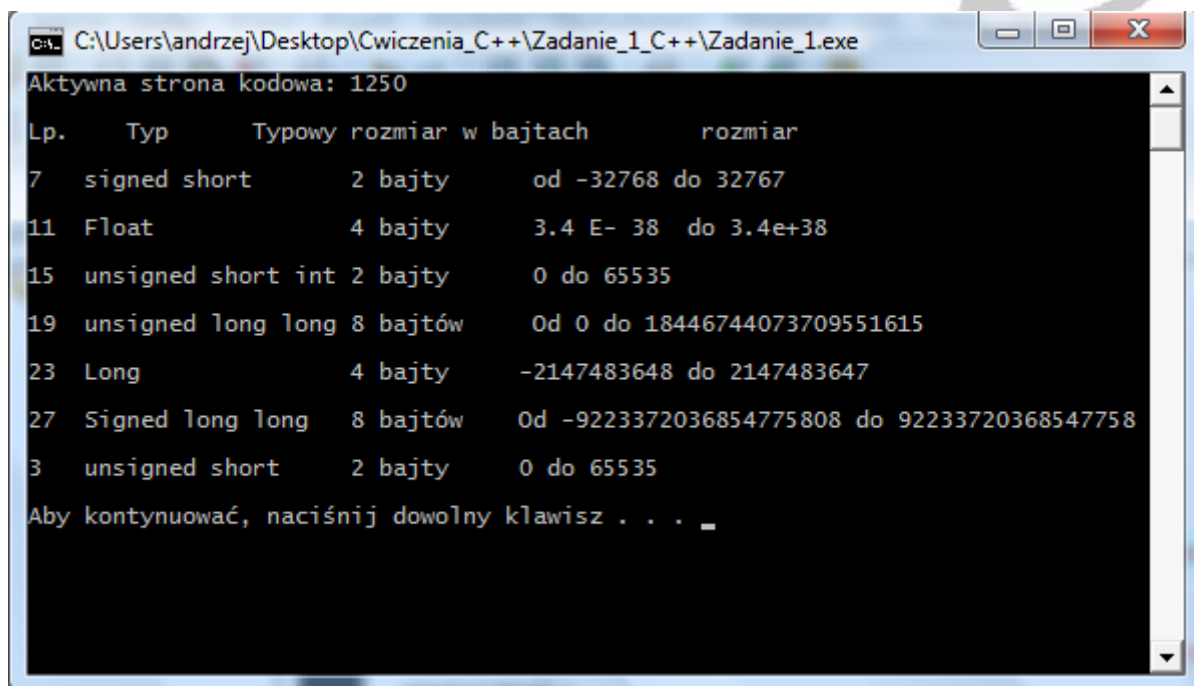
int main(int argc, char *argv[])
{
    printf("Autor: .....");
    cout<<endl;
    printf("TO JA, TWOJ PROGRAM-PIERWSZY.CPP");
    cout<<endl;
    return 0;
}
```

Uzupełnij pole autor.

Dodaj kodowanie polskich znaków.

Zadanie 1.

Napisz program opisujący zakresy wartości przedstawionych w tabeli (Zakres wartości typów danych – cz.2) umieszczonej w instrukcji w formie tabelarycznej (bez ramek). Wpisujesz z tabeli numery wierszy nr_z_dziennika+4. Uczeń o numerze 7 w dzienniku wpisuje 7, 11, 15, 19, 23, 27, 3.



Zadanie 2.

Napisz program, który wywoła pięć dowolnych funkcji DOS. Przed wywołaniem funkcji na monitorze pojawi się napis:

Np.

Teraz kasuję ekran i dalsza część programu po naciśnięciu klawisza Enter

Zadanie 3.

Wykorzystując przykład 3 i 4 napisz program, który narysuje następującą tabelę:

!	kolor	!	numer koloru	!
!	wszyst kolory	!	!
!	biały	!	15 oraz 7	!
!	żółty	!	14 oraz 6	!
!	purpurowy	!	13 oraz 5	!
!	czerwony	!	12 oraz 4	!
!	jasnoniebieski	!	11 oraz 3	!
!	zielony	!	10 oraz 2	!
!	ciemnoniebieski	!	9 oraz 1	!
!	szary	!	8 oraz 0	!

Każda linia (od drugiej) pisana takim kolorem, jaki opisuje. Kolory od 0 do 15.
Pause→białym

```

C:\Users\andrzej\Desktop\Cwiczenia_C++\Zadanie_3_C++\Za
Aktywna strona kodowa: 1250
+-----+
!      kolor  !  numer koloru  !
+-----+
! wszystkie kolory !  .....  !
+-----+
!      biały   !  15 lub 7   !
+-----+
!      żółty   !  14 oraz 6   !
+-----+
!  purpurowy   !  13 oraz 5   !
+-----+
!  czerwony    !  12 oraz 4   !
+-----+
!  jasnoniebieski !  11 oraz 3   !
+-----+
!   zielony     !  10 oraz 2   !
+-----+
!  ciemnoniebieski !  9 oraz 1   !
+-----+
!   szary       !  8 oraz 0   !
+-----+

```

Zadanie 4.

Wykonaj rysunek choinki noworocznej(gałęzie, stojak, pień, bombki , świeczka) . Z użyciem kolorów oraz funkcji gotoxy(int x, int y). Wpisz funkcję gotoxy nad main().

Uwaga:

Gdy chcesz wprowadzić znak / na ekran konsoli w CPP musisz zapisać dwa razy // ponieważ znak / jest znakiem specjalnym.

Czyli napisz

```
Cout<<"//";
```

Przykład 3

Określanie kolorów tła i liter z użyciem <windows.h>

```
#include <iostream>
#include <windows.h>

using namespace std;

void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

int main()
{
    cout << "Tekst nie kolorowany\n\n";
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
    FOREGROUND_GREEN | FOREGROUND_INTENSITY);
    cout << "Tekst pokolorowany\n\n";
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15|
    FOREGROUND_INTENSITY);
    // 15 to kolor biały
    cout << "Tekst nie kolorowany\n\n";
    gotoxy(10,10);
    cout<<"K";
    cout<<endl;
    return 0;
}
```

Przykład 4

```
#include <windows.h>
#include <iostream>
#include <stdlib.h>

using namespace std;

int main()
```

```

{

HANDLE hOut;

hOut = GetStdHandle(STD_OUTPUT_HANDLE);
cout << "Standart" << endl << endl;

SetConsoleTextAttribute(hOut,BACKGROUND_RED);
cout << "czerwony." << flush << endl << endl;

SetConsoleTextAttribute(hOut,FOREGROUND_GREEN);
cout << "zielony." << endl << endl;

SetConsoleTextAttribute(hOut,FOREGROUND_BLUE);
cout << "niebieski." << endl << endl;

system("PAUSE");
return 0;
}

```

Przykład 4A

Wyjaśnienie działania funkcji gotoxy:

```

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

void gotoxy(int x, int y ); // Deklaracja funkcji, informujemy kompilator, że taka funkcja
istnieje

using namespace std;

int main()
{
    cout << "Działanie funkcji gotoxy - punkt początkowy (0,0)";
    gotoxy(15,5);    /* Odwołujemy się do funkcji gotoxy z dwoma parametrami, 15 i 5
po wykonaniu kodu znajdującego się w gotoxy kursor przemieści się do
punktu (15,5)*/
    cout << " Nowy punkt (15,5) - Teraz kursor konsoli jest tutaj";
    getch(); // a to jest oczekiwanie na wciśnięcie jakiegos przycisku
}
/* void można tłumaczyć jako pusty, co skutkuje tym że funkcja w przeciwieństwie do int
nie zwraca żadnej wartości. W środku deklarujemy 2 zmienne x i y do których przekazujemy
2 wartości w funkcji main za pomocą gotoxy(15,5) pierwsza liczba w odwołaniu będzie
zapisana jako x a druga jako y */

void gotoxy( int x, int y )
{

```

```
COORD coord; // tutaj deklarujemy strukture coord, i zmienna coord będzie mogła
przechowywać współrzędne, tutaj 2 zmienne X i Y
coord.X = x; // tutaj do zmiennej coord.X zapisujemy nasze x w naszym przypadku jest to
15
coord.Y = y; // tak samo jak wyżej tylko do coord.Y i tutaj jest to 5

SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord );
/* pierwszy parametr to uchwyt do urządzenia do którego się odwołuje czyli w tym
przypadku jest to bufor konsoli a drugi parametr to współrzędne nowej pozycji kursora. */
}
```



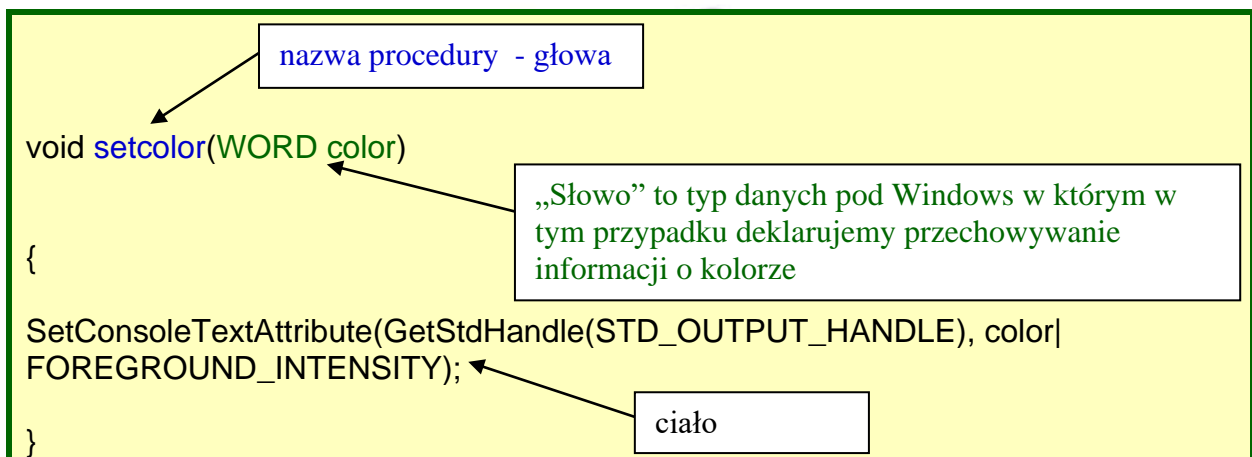
Jak podczas rozwiązywania zadań ułatwić sobie życie:

Przy rozwiązywaniu zadań pomocne mogą się okazać następujące informacje:

\n – powoduje przejście do następnej linii;

\t – wstawia tabulator;

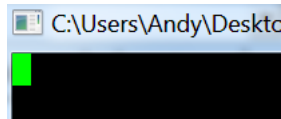
Możemy też ułatwić sobie zadanie tworząc void-a, który skróci nam bardzo kod w programie:



teraz wystarczy wywołać procedurę `setcolor` podając jej odpowiedni parametr koloru np.:

```
setcolor(14);  
cout<<" żółty "<<endl;  
setcolor(13);  
cout<<" purpurowy "<<endl;
```

możemy też wywołać znak za pomocą numeracji kodu ASCII:
cout<<(char) 219;



Część 2

Zmienne, stałe, obliczenia, formatowanie wydruków.

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

Omów jakie nazwy mogą (nie mogą) mieć zmienne i stałe.

Pytanie 2

wymień oraz, krótko opisz sześć podstawowych typów danych wraz z rozmiarem w bajtach..

Pytanie 3

wymień uwagi na temat stosowania nazw,(wypisz 10 słów kluczowych, które nie mogą być nazwami)

Pytanie 4

Omów:

- deklarację,
- inicjację,
- definicje zmiennych

Pytanie 5

narysuj schemat ogólnego podziału danych

Pytanie 6

omów podstawowe modyfikacje do podstawowych danych,

Pytanie 7

omów deklarację stałych na przykładzie

Pytanie 8

- a) Omów sposoby zapisu liczb zmiennoprzecinkowych wraz z przykładem.
- b) Zamień liczbę z postaci wykładniczej na numer_w_dzienniku.miesiąc_urodzeniae4 na liczbę bez potęgi.(tam jest kropka)

Pytanie 9

W jakim miejscu programu można dokonwać deklaracji zmiennych w C++..

Pytanie 10

Omów instrukcję cin. Podaj przykład jednej instrukcji cin do wczytania dwóch zmiennych.

Pytanie 11

Omów instrukcję przypisania w C++.

Pytanie 12

Zapisz co oznacza zapis w treści programu \n

Pytanie 13

Opisz Instrukcje **cin** wraz z przykładem oraz przykład wczytania dwóch zmiennych o nazwach **dwie_pierwsze_litery_nazwiska** i **dwie_pierwsze_litery_imienia**.

Pytanie 14

Opisz inkrementację oraz dekrementację wraz przykładem.

Pytanie 15

Przerysuj tabelę operatorów arytmetycznych języka C++.

Pytanie 16

Przerysuj tabelę Operatorów relacji języka C++.

Pytanie 17

Zapisz Operatory logiczne języka C++ wraz z przykładem.

Pytanie 18

Omów funkcję **scanf()** (Znaczenie, składnia bez opisu, przykład1 z opisem, przykład 2 z opisem)

Pytanie 19

Omów funkcję **printf()** (Znaczenie, składnia bez opisu, przykład1 z opisem, przykład 2 z opisem)

Pytanie 20

Zanotuj trzy wzorce konwersji dla postać %s, %d, %f

Pytanie 21

Opisz trzy zakresy ważności zmiennych.

Pytanie 22

Opisz przykrywanie zmiennych.

Nazwy zmiennych i stałych

W nazywaniu zmiennych masz bardzo dużą dowolność. Nadając zmiennej nazwę podczas jej tworzenia (deklarowania) możesz użyć dowolnej kombinacji liter i cyfr oraz znaku podkreślenia z trzema zastrzeżeniami:



- nie możesz używać spacji,
- nie możesz używać znaków specjalnych (np. ?, !, /, #, \$ itp). Polskie litery: ą, ę, ć, ł, ó, ś, ż, ź - to także znaki specjalne!
- nie możesz użyć cyfry na pierwszym miejscu.

Pozostałe uwagi dotyczące stosowania nazw:

- nazwa powinna być znacząca np. **promien** a nie nic znacząca nazwa **mien**,
- nazwa może się składać z dużych i małych liter lecz musimy pamiętać, że kompilator rozróżnia duże i małe litery,
- przyjęto, że nazwy zmiennych piszemy małymi literami a stałych dużymi,
- nie można stosować nazw, które są zarezerwowane dla słów kluczowych:
asm , auto, break, case, char, class, const, continue, default, delete, do, double, else, enum, extern, float, for, friend, goto, huge, if, inline, int, interrupt, long, near, new, operator - operator, pascal, private, protected, public, register, return , short, signed, unsigned , sizeof, static, struct, switch, this, typedef, union, virtual, void, volatile, while;

Tak więc, tworząc (deklarując) zmienne, pamiętaj, że, **język C++ uwzględnia wielkość liter**. Oznacza to, że zmienna o nazwie *data* jest, z punktu widzenia komputera, czymś zupełnie różnym od zmiennej o nazwie *Data* lub jeszcze innej zmiennej o nazwie *DATA*.

Możesz zatem utworzyć trzy różne zmienne:

```
int data;
int Data;
int DATA;
```

choć byłoby to niepraktyczne i nieczytelne. Aby uniknąć takich sytuacji, programiści przyjmują różne **konwencje nazywania zmiennych**.

Jedną z najpopularniejszych konwencji jest używanie w nazwach zmiennych wyłącznie małych liter. Jeżeli w nazwie zmiennej występują dwa wyrazy (np. "dzień tygodnia") stosuje się dwa sposoby:

1. użycie podkreślnika - `dzien_tygodnia`,
 2. użycie dużej litery na początku drugiego wyrazu - `dzienTygodnia`.
- Tę drugą formę nazywa się **zapisem wielbłąda**.

Deklaracja, inicjacja, definiowanie zmiennych.

DEKLARACJA

np. **int a** oznacza mniej więcej, jakbyśmy powiedzieli kompilatorowi: “Jakbyś gdzieś spotkał nazwę **a** to wiedz, że jest to zmienna typu **int (integer)**”. Określa typ wartości, jakie może przechowywać dana stała lub zmienna.

INICJACJA

to pierwsze przypisanie stałej lub zmiennej. Powoduje przydzielenie pamięci.

DEFINICJA

to jednocześnie deklaracja i inicjacja.

Rasumując: definicja funkcji, to wszystko to, co jest zawarte pomiędzy nawiasami klamrowymi { } występującymi po deklaracji funkcji. Deklaracja funkcji to ta linijka gdzie funkcja jest tworzona, tzn. gdzie jest podany typ wartości, którą funkcja zwraca, nazwa funkcji oraz w nawiasach okrągłych () parametry, które do funkcji przekazujemy.

Język C/C++ operuje sześcioma podstawowymi typami danych:

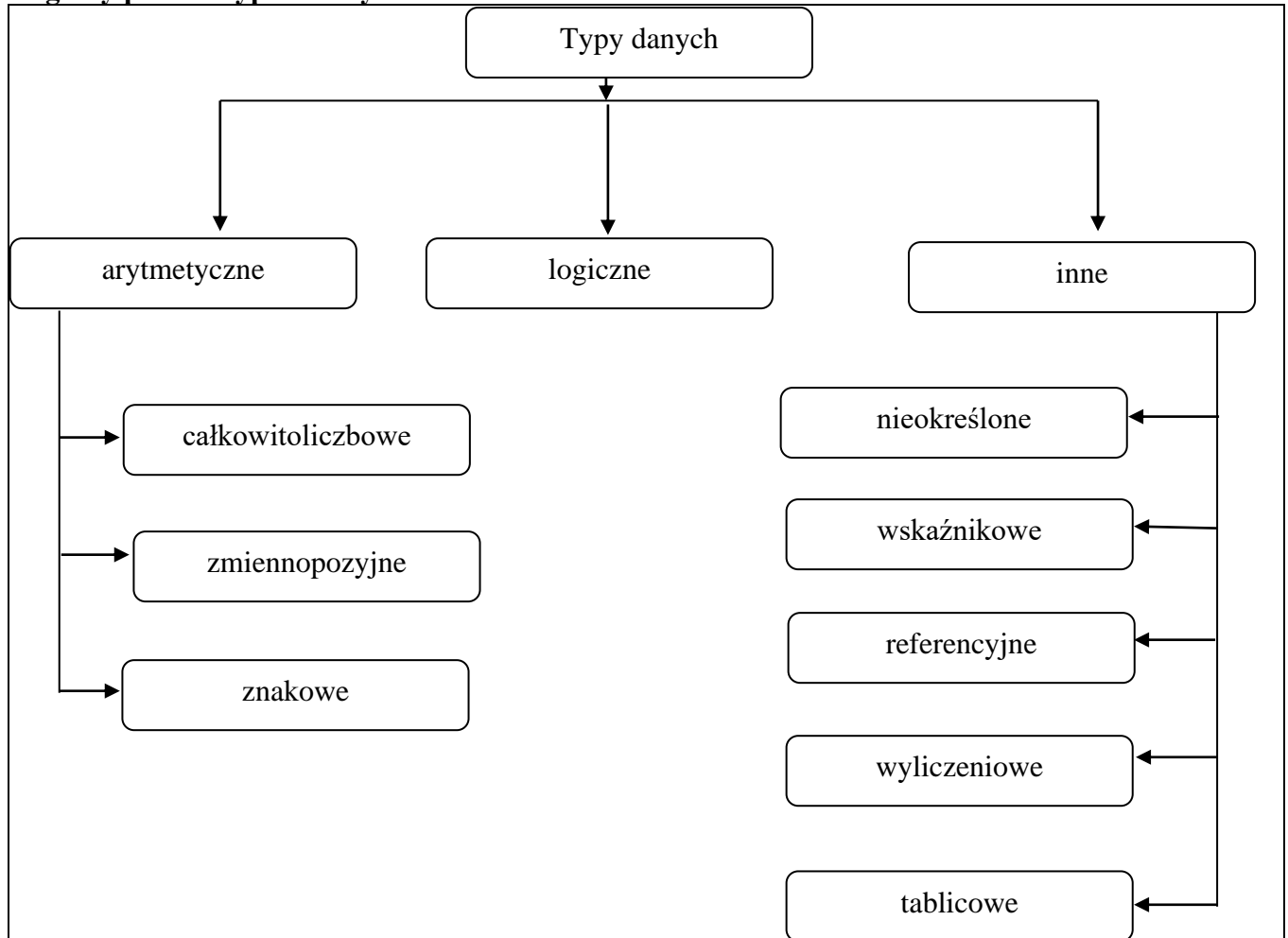
- char (znak, numer znaku w kodzie ASCII) - 1 bajt;
- int (liczba całkowita) - 2 bajty;
- float (liczba z pływającym przecinkiem) - 4 bajty;
- double (podwójna ilość cyfr znaczących) - 8 bajtów;
- bool (wartość logiczna)
- void (nieokreślona) 0 bajtów.

Podstawowe typy danych mogą być stosowane z jednym z czterech modyfikatorów:

- signed / unsigned - ze znakiem albo bez znaku
- long / short - długi albo krótki

Zakres wartości przedstawiono w Tabeli poniżej.

	Typ	Typowy rozmiar w bajtach	rozmiar
1	char signed	1 bajty	-128 do 127
2	unsigned char	1 bajty	0 do 255
3	unsigned short	2 bajty	0 do 65535
4	Int	2 bajty	-32768 do 32767
5	unsigned int	4 bajty	0 do 65535
6	long int	4 bajty	-2147483648 do 2147483647
7	signed short	2 bajty	od -32768 do 32767
8	signed int	4 bajty	Od -2147483648 do 2147483647
9	unsigned long int	4 bajty	0 do 4294967295
10	signed long int	4 bajty	-2147483648 do 2147483647
11	Float	4 bajty	3.4 E- 38 do 3.4e+38
12	Double	8 bajtów	1.7 E-308 do 1.7 E+308
13	long double	10 bajtów	1.2 E-4932 do 1.2E+4932
14	Bool	1	true, false
15	unsigned short int	2 bajty	0 do 65535
16	Unsigned	4 bajty	Od 0 do 4294967295
17	unsigned long	4 bajty	Od 0 do 4294967295
18	unsigned long long int	8 bajtów	Od 0 do 18446744073709551615
19	unsigned long long	8 bajtów	Od 0 do 18446744073709551615
20	Short	2 bajty	Od -32768 do 32767
21	Short int	2 bajty	Od -32768 do 32767
22	Signed	4 bajty	Od -2147483648 do 2147483647
23	Long	4 bajty	-2147483648 do 2147483647
24	Signed long	4 bajty	-2147483648 do 2147483647
25	Long long	8 bajtów	Od -9223372036854775808 do 9223372036854775807
26	Long long int	8 bajtów	Od -9223372036854775808 do 9223372036854775807
27	Signed long long	8 bajtów	Od -9223372036854775808 do 9223372036854775807
28	Signed long long int	8 bajtów	Od -9223372036854775808 do 9223372036854775807

Ogólny podział typów danych:**Stale**

Stała to taka zmienna, której wartość można przypisać tylko raz.

```
const float PI = 3.14159;
```

nie można przypisać w programie żadnej innej wartości, innymi słowy zapis: jest jednocześnie DEKLARACJĄ, DEFINICJĄ i ZAINICJOWANIEM stałej PI.

Przykład :

float a,b,c;	(DEKLARACJA)
const float euler = 2.7	(DEFINICJA)
y = 441;	(ZAINICJOWANIE zmiennej)

Sposoby zapisu liczb zmiennoprzecinkowychSposób 1

zapis części ułamkowej pisanej po kropce np. 13.345

Sposób 2

sposób wykładniczy $34.6e4 = 34.6 * 10^4 = 34.6 * 10000 = 346000$

Przykład 5**Temat:**

Przykład demonstruje definiowanie zmiennych, wczytywanie ich wartości z klawiatury oraz operacje na nich. Całość z użyciem strumieni.

Wykonaj:

- Wpisz temat do zeszytu
- Wpisać Przykład 5 (łącznie z komentarzem) do komputera. Nazwa pliku na dysku p5_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.

```
/* Oto program pokazujący operacje wczytywania z klawiatury i obliczania wartości
wyrażenia */

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

using namespace std;

int main(int argc, char *argv[])
{
    float metry;
    float kilometry;
    float k=1000;

    cout<<"Podaj liczbe metrow:";

    cin>>metry;                //wczytanie z klawiatury

    kilometry=metry/k;

    cout<<"Oto wynik przekształcenia:\n"<<metry<<"  metrow to  "<<kilometry<<"
kilometrow.";
    cout<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Wy tłumaczenie instrukcji użytych w przykładzie powyżej.

Definicje zmiennych*Pierwsza zmienna:*

Instrukcja **float metry** ; powoduje, że tworzymy zmienną o nazwie **metry**, która jest typu **float**. Typ ten służy do przechowywania liczb zmiennoprzecinkowych czyli takich, które mają przecinki.

Druga zmienna:

Tworzymy zmienną **kilometry**, która jest typu **float**.

Trzecia zmienna:

Definiujemy w niej zmienną k typu float i **od razu wpisujemy do niej wartość 1000**. Uwaga: W momencie utworzenia zmiennych tzw. lokalnych znajduje się w nich wartość przypadkowa.

W C++ można definiować zmienne wtedy, kiedy się zorientujemy, że są nam potrzebne.

Opis instrukcji

cin >> cośćam ; oznacza wczytanie z klawiatury do zmiennej **cośćam**. Jak widać, jest to ruch tak jakby odwrotny do wypisywania – tak nam sugerują strzałki.

Przeliczenia na kilometry. Dzielenie to /.

Po przeliczeniu wypisujemy na ekran wynik.

'\n' – nowa linia

Instrukcja cin → wczytywanie (pobierania) danych do programu z klawiatury lub pliku zewnętrznego.

np. cin>>promien;

Wczytanie z klawiatury liczby do zmiennej o nazwie **promien**.

np. cin>>x>>y

Wczytanie z użyciem jednej instrukcji cin dwóch zmiennych o nazwach **x** i **y**.

Operacja przypisania

do przypisania używa się znaku = a nie := jak w pascalu.

np.

p = (a+b)/2*h;

x = y = 10; przypisanie zmiennym x i y wartości 10

Inkrementacja

Pojęcie to oznacza zwiększanie o jeden wartości zmiennej

np.

x++

wartość zmiennej x została zwiększona o jeden

Dekrementacja

Pojęcie to oznacza zmniejszanie o jeden wartości zmiennej

np.

y--

wartość zmiennej y została zmniejszona o jeden

OPERATORY ARYTMETYCZNE języka C++.

Operator	Nazwa	Tłumaczenie	Działanie
+	ADDition	Dodawanie	Suma liczb
-	SUBstraction	Odejmowanie	Różnica liczb
*	MULTiplication	Mnożenie	Iloczyn liczb
/	DIVision	Dzielenie	Iloraz liczb
%	Moduluj	Dziel Modulo	Reszta z dzielenia

Operatory relacji języka C++.

Operator	Nazwa	przykład	wynik
==	Równe	34==(37-3)	prawda
!=	różne od	9!=(15-6)	Fałsz
<	Mniejsze	10<11	prawda
>	Większe	9>8	fałsz
<=	mniejsze lub równe	7<=7	prawda
>=	większe lub równe	3>=4	fałsz

Operatory logiczne języka C++.

AND → && koniunkcja **i**
 OR → || alternatywa **lub**
 NOT → ! zaprzeczenie **nie**

np.

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

Zadanie 5.

Wykonaj program działania kalkulatora, który po wczytaniu dwóch liczb naturalnych dodatnich w jednej instrukcji poda wynik:

- Suma liczb
- Różnica liczb
- Iloczyn liczb
- Iloraz liczb
- Reszta z dzielenia

Zmienne a i b zapisz z trzema literami nazwiska np.: a_raw.

Zmienne typu Int oprócz zmiennej iloraz, ta zmienna typu float.

Zapisz wzór na iloraz:

Iloraz=1.0*a_raw/b_raw;

Wygląd ekranu:

Np.

Podaj a=

Podaj b=

Wynik działania programu(wygląd ekranu):

suma=5

różnica=-1

iloczyn=6

iloraz=0,66

reszta z dzielenia=2

```
Podaj liczbę a: 2
Podaj liczbę b: 3
suma= 5
różnica= -1
iloczyn= 6
iloraz= 0.666667
reszta z dzielenia= 2
```

Zadanie 6.

Wykonaj program zamiany radianów na stopnie po wczytaniu z klawiatury radianów.

Użyj zdefiniowanej stałej PI w programie. Wypisz jednostki przy wyprowadzaniu wyników np. stopnie lub radiany.

Użyj zmiennych z trzema literami nazwiska.

Dane do sprawdzenia

Radiany=3.14159 otrzymasz wynik stopnie=180

```
Podaj wartość radianów: 3.14159
3.14159 rad = 180 stopnie
```

Zadanie 7.

Wykonaj program zamiany stopni na radiany po wczytaniu z klawiatury stopni.

Użyj zdefiniowanej stałej $PI = 3.14159$ w programie. Wypisz jednostki przy wyprowadzaniu wyników np. stopnie lub radiany.

Użyj zmiennych z trzema literami nazwiska.

Dane do sprawdzenia

stopnie=180 otrzymasz wynik Radiany=3.14159

```
Podaj wartość stopni: 180
180 stopni = 3.14159 rad
```

funkcja scanf()

Znaczenie:

Funkcja scanf() wczytuje dane dowolnego typu i dowolnego rozmiaru (także łańcuchy znakowe).

Składnia funkcji scanf():

#include<stdio.h> // taki plik nagłówkowy musisz dołączyć do programu;

int scanf(const char *format, [adres, ...]);

Opis:

- Zmienna **format** reprezentuje łańcuch formatujący (format string), w którym mogą być użyte, różne specyfikatory formatu (do wczytywania danych różnych typów).
- Jeśli działanie funkcji zakończy się poprawnie, funkcja zwraca **liczbę wczytanych danych** w przeciwnym razie zwraca **EOF**.
- Wielokropek** oznacza, że funkcja scanf() jest funkcją o zmiennej liczbie argumentów, tzn. można nią wczytać dowolną ilość danych.
- Funkcja **wczytuje dane** do pojawienia się jednego ze znaków: **spacja, ENTER, TAB, tabulacja pionowa, FF (znak końca stronicy)**.

Przykład z omówieniem→Wywołanie funkcji scanf():

Przykład 1

```
scanf("%d %f",&x,&y);
```

Opis:

Funkcja scanf() wczytuje dwie wartości typu int i float podstawiane pod adres zmiennych x i y (przy użyciu operatora adresowego &).

Przykład 2

```
scanf("%s",str);
```

Opis:

Wczytuje łańcuch znaków (specyfikator %s) i zapisuje go pod adres tablicy str (nie ma operatora &, bo nazwa tablicy jest wskaźnikiem - wskazuje adres pierwszego elementu) i dodaje na końcu wczytanego łańcucha znak '\0'. Wszystkie zmienne trzeba najpierw zadeklarować.

funkcja printf()

Znaczenie:

Służy do wyprowadzania danych różnych typów na ekran.

Składnia funkcji scanf():

Składnia funkcji printf():

```
#include<stdio.h> // taki plik nagłówkowy musisz dołączyć do programu;
int printf(const char *format_string, ...);
```

Opis:

- Pierwszy argument format_string to łańcuch znaków zawierający specyfikatory formatu (%d, %c itp. - patrz specyfikatory formatu).
- Wielokropek oznacza, że jest to podobnie jak scanf() funkcja o zmiennej liczbie argumentów, w tym miejscu
- znajduje się dowolna ilość danych wyprowadzanych na ekran. Ważne, aby ich liczba zgadzała się z liczbą specyfikatorów formatu. Jeśli działanie funkcji zakończy się poprawnie funkcja zwraca liczbę wyprowadzanych danych, w przeciwnym razie zwraca EOF.

Przykład z omówieniem → Wywołanie funkcji printf():

Przykład 1

```
printf("suma dwóch liczb %d + %d = %d\n",x,y,suma);
```

Opis

Funkcja printf() drukując na ekranie łańcuch znakowy (ten pomiędzy cudzysłowami (" ")) w miejsce specyfikatorów %d wstawia odpowiednio dane znajdujące się po przecinku: x, y, suma.

Przykład 2

```
printf("pierwiastek z liczby %f = %f\n",a,sqrt(a));
```

Opis

Funkcja natomiast w miejsce drugiego specyfikator %f wstawia wartość zwracaną przez funkcję sqrt() (funkcje matematyczne). Tak więc argumentem funkcji printf() mogą być nie tylko zmienne ale także dowolne funkcje, czy wyrażenia.

W łańcuchu formatującym funkcji printf() znalazł się także znak specjalny '\n', oznaczający przejście do następnej linii, więc funkcja ta może zawierać dowolne znaki specjalne (ich spis znajdziesz tutaj).

Wzorce konwersji

mają postać %s, %d, %f

a)

%s - wyprowadź łańcuch znaków (s - String - łańcuch)

Przykład:

```
printf("%s","jakiś tekst");
```

format "%s" jest formatem domyślnym dla funkcji printf().

Przykład:

```
printf("%35s", "jakiś tekst");
```

spowoduje uzupełnienie napisu spacjami do zadanej długości 35 znaków. Funkcja printf() operuje tzw. POLEM WYJŚCIOWYM zwanym inaczej matrycą. Długość pola wyjściowego możemy określić przy pomocy liczb wpisanych pomiędzy znaki % oraz typ - np. s.

b)

%c - wyprowadź pojedynczy znak (c - Character - znak)

Przykład:

```
printf("%c", 'X');
```

spowoduje wydrukowanie litery X)

c)

%d - wyprowadź liczbę całkowitą typu int w postaci dziesiętnej → d - Decimal - dziesiętny.

Przykład:

```
printf("%d", 1994);
```

d)

%f - wyprowadź liczbę rzeczywistą typu float w postaci.

Możemy także określić ilość cyfr przed i po przecinku (f - Floating point - zmienny przecinek).

Przykład:

```
printf("%f", 3.1416);
```

```
printf("%3.2f", 3.14159);
```

e)

%o - wyprowadź liczbę całkowitą typu int w postaci ósemkowej o - Octal - ósemkowa.

Przykład:

```
printf("%o", 255);
```

f)

%x - wyprowadź liczbę całkowitą typu int w postaci szesnastkowej x - heXadecimal - szesnastkowa.

%x lub %X - cyfry szesnastkowe a,b,c,d,e,f lub A,B,C,D,E,F.

g)

%ld - liczba całkowita "długa" - long int.

h)

%Lf - liczba rzeczywista poczwórnej precyzji typu long double float.

i)

%e - liczba w formacie wykładniczym typu 1.23e-05 (0.0000123)

j)

%g - automatyczny wybór formatu %f albo %e.

Uogólnijmy sposób zastosowania wzorca

formatu: %[przełączniki][szerokość_pola][.precyzja][rozmiar]Typ

Przykład.A. wyprowadzania złożonych napisów.

```
printf("Iloczyn 3 %c 5 %8s %d", '*', "wynosi ", 15);
```

Wytłumaczenie:

"Iloczyn_3_" - funkcja wyprowadza łańcuch znaków.

%c – funkcja napotyka specyfikator i wstawi pierwszy napotkany znak po przecinku

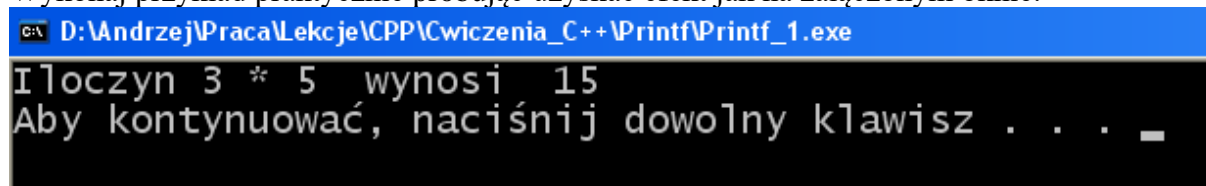
tutaj wyprowadzi pojedynczy znak, czyli '*'.

`_5_` - dalej wyprowadza jako łańcuch znaków.

`%8s` - funkcja napotyka kolejny specyfikator - wyprowadza łańcuch znaków umieszczony po przecinku "wynosi_" uzupełniając go z przodu spacjami do długości 8 znaków.

`%d` - funkcja napotyka trzeci kolejny specyfikator - wyprowadza 15 jako liczbę dziesiętną.

Wykonaj przykład praktycznie próbując uzyskać efekt jak na załączonym oknie:



```
D:\Andrzej\Praca\Lekcje\CPP\Cwiczenia_C++\Printf\Printf_1.exe
Iloczyn 3 * 5 wynosi 15
Aby kontynuować, naciśnij dowolny klawisz . . . _
```

Język C pozwala na wstawienie w specyfikatorze formatu liczby pomiędzy znak % z literę. I tak:

pojedyncza liczba oznacza minimalną szerokość pola wyjściowego (ilość znaków), dzięki czemu wyprowadzone znaki mogą zajmować jak najmniej miejsca, np.:

```
printf("%6d \n",12);
```

funkcja printf() wyprowadzi liczbę 12 w polu o szerokości 6 znaków z wyrównaniem do prawej.

umieszczenie przed liczbą znaku '-' spowoduje wyrównanie wyprowadzonych znaków do lewej, np.:

```
printf("%-7d \n",12);
```

liczba 12 wyprowadzona zostanie w 7-io znakowym polu z wyrównaniem do lewej (domyślnie wyrównanie jest do prawej).

po liczbie (lub bez niej) można dodać znak kropki '.' i następną liczbę. Połączenie znak + kropka daje specyfikator precyzji:

dla liczb zmiennoprzecinkowych:

```
printf("%-10.3f \n",12.34567);
```

wyprowadzona zostanie liczba 12.346 na 10-io znakowym polu z 3 liczbami po przecinku (zaokrąglonymi) z wyrównaniem do lewej.

dla liczb całkowitych i łańcuchów tekstowych:

```
printf("%10.6d \n",123);
```

wyprowadzona zostanie liczba 000123 w 10-io znakowym polu na 6-iu pozycjach (jeśli dana liczba zajmuje mniejszą ilość znaków niż 6 wolne miejsca są wypełniane zerami). Liczba po kropce to maksymalna szerokość pola wyjściowego.

Przykład 6

Temat:

Przykład na formatowanie i wyprowadzanie danych.

Wykonaj:

- Wpisz temat do zeszytu

- Wpisać Przykła6 (łącznie z komentarzem) do komputera. Nazwa pliku na dysku p6_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.

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

using namespace std;

int main(int argc, char *argv[])
{
    float x,y;
    float wynik;

    printf("Zamieniam ulamki zwykłe na dziesiętne\n");
    printf("\nPodaj licznik ulamka:");
    scanf("%f",&x); /*pobiera liczbe z klawiatury*/
    printf("\nPodaj mianownik ulamka:");
    scanf("%f",&y);

    wynik=x/y; /*tu wykonuje sie dzielenie*/

    printf("\n%f: %f=%f", x,y, wynik);
    cout<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Przykład 7

Temat:

Przykład na skomplikowane formatowanie i wyprowadzanie danych.

Wykonaj:

- Wpisz temat do zeszytu
- Wpisać Przykła7 (łącznie z komentarzem) do komputera. Nazwa pliku na dysku p7_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.

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

int main(int argc, char *argv[])
{
```

```

printf("Skomplikowany napis:\n");
printf("Iloraz 135 %c 5 %8s %d", ':', "wynosi ", 27);
printf("\n");
system("PAUSE");
printf("\nWyrażenie jako argument:\n");
printf("Iloraz 135 %c 5 %9s %d", ':', "wynosi ", 135/5);
printf("\n\n");
printf("Przyjrzyj się i naciśnij klawisz...");
printf("\n");
system("PAUSE");
return EXIT_SUCCESS;
}

```

Zadanie 8.

Napisz program, który po wczytaniu liczby x poda tę liczbę w formie szesnastkowej w następującej formie:

np. . dla wczytanej zmiennej x równej tysiąc otrzymamy wygląd ekranu →

wczytana liczba $x=1000$

liczba w formie szesnastkowej $x=3e8$

```

C:\Users\andrzej_r\Desktop\Pulpit\Cwiczenia_C++\Zadanie_8_C++\Zadanie_8.exe
Aktywna strona kodowa: 1250
Wprowadzam liczę x w postaci dziesiętnej
Podaj liczbę x: 1000
wczytana liczba= 1000 liczba w formie szesnastkowej wynosi= 3e8
liczba w formie ósemkowej wynosi= 1750
Aby kontynuować, naciśnij dowolny klawisz . . .

```

Użyj funkcji `printf` oraz `scanf`.

Pamiętaj, że zmienna „ x ” musi być całkowita (taki format wczytywania danych `%d`).

Nazwy zmiennych tak, aby zawierały trzy pierwsze litery nazwiska.

Jak wykonać szukaj w tej instrukcji np., jaka zmienić na ósemkowy?

Zadanie 9.

Uwagi:

a) Użyj funkcji `printf` oraz `scanf`

b) Nazwy zmiennych tak, aby zawierały trzy pierwsze litery nazwiska.

Napisz program, który po wczytaniu liczby x (naturalnej) poda wynik działania programu w następującej formie:

np. dla wczytanej zmiennej x równej trzy otrzymamy wygląd ekranu →

wczytana liczba $x=3$

kwadrat liczby $x=3$ $3^2=9$

sześcian liczby $x=3$ $3^3=27$

```

Podaj liczbę naturalną: 3
wczytana liczba x= 3
kwadrat liczby x= 3 3^2=9
sześcián liczby x= 3 3^3=27
Aby kontynuować, naciśnij dowolny klawisz . . .

```

Przykład 8

Temat:

Program oblicza pole oraz obwód trójkąta po podaniu kąta (w stopniach) i przeciwprostokątnej.

Wykonaj:

Wpisz temat do zeszytu

- Wpisać Przykła8 (łącznie z komentarzem) do komputera. Nazwa pliku na dysku p8_cztery_pierwsze_litery_nazwiska
- Przepisać treści programu do zeszytu.
- Narysuj rysunek trójkąta prostokątnego w zeszycie oznacz boki a b c oraz kąt alfa. Innym kolorem zaznacz te zmienne na rysunku, które są danymi w zadaniu. Pod rysunkiem zapisz wzory na zamianę stopni na radiany.

```

#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;//definicja stalej
    float a,b,c;
    float pole,obwod,alfa_x,alfa_s;
    printf("Umiem obliczac pole oraz obwod trojkata\n");
    printf("po podaniu kata i przeciwprostokatnej\n");
    printf("podaj przeciwprostokatna\n");
    scanf("%f",&c);
    printf("podaj kat w stopniach\n");
    scanf("%f",&alfa_s);
    printf("wczytane dane");
    printf("\nc=%.2f",c);
    // w linii wyżej definiowane są dwa miejsca po przecinku
    printf(" cm");
    printf("\nalfa=%f %s\n",alfa_s,"st");
    alfa_x=alfa_s*PI/180;
    a=c*cos(alfa_x);
    b=c*sin(alfa_x);

```



```

pole=a*b/2;
obwod=a+b+c;
printf("wyniki");
printf("\na=%f",a);
printf(" cm");
printf("\nb=%f",b);
printf(" cm");
printf("\npole=%f",pole);
printf(" cm^2");
printf("\nobwod=%f",obwod);
printf(" cm");
cout<<endl;

system("PAUSE");
return EXIT_SUCCESS;
}

```

ZADANIE 10.

Nazwy zmiennych tak, aby zawierały trzy pierwsze litery nazwiska.

Napisz program obliczający przy stałej przeciwprostokątnej $c=30$ cm (wpisz jako stałą) i wczytywanej z klawiatury przyprostokątnej a :

- Narysuj rysunek trójkąta prostokątnego w zeszycie oznacz boki a b c oraz kąt α . Innym kolorem zaznacz te zmienne na rysunku, które są dane w zadaniu. Pod rysunkiem zapisz wzory na b , pole, \cos_alfa , \sin_alfa , tg_alfa .

Oblicz

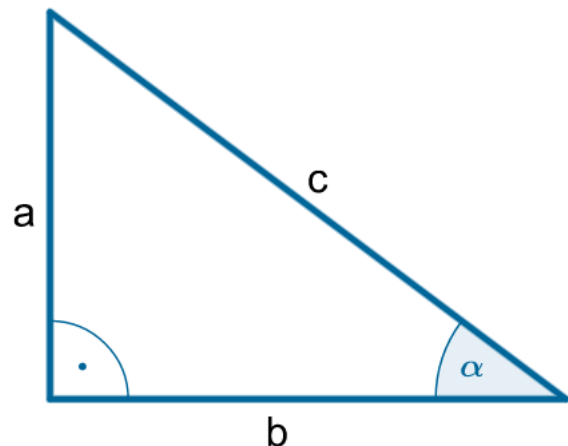
- nieznaną bok trójkąta b ;
- pole trójkąta;
- cosinus, sinus, tangens dowolnego kąta ostrego
- kąty trójkąta wyrażone w stopniach

Przy wynikach pisz jednostki, formatuj wydruki do trzech miejsc po przecinku.

$$\sin \alpha = \frac{a}{c}, \cos \alpha = \frac{b}{c}, \operatorname{tg} \alpha = \frac{a}{b}$$

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha}$$

$$a^2 + b^2 = c^2$$



Do obliczenia kątów użyj instrukcji $\operatorname{atan}(x)$ np. $ALFA = \operatorname{atan}(1)$ to $ALFA = \pi/4$ czyli $ALFA = 45$ stopni. Wykorzystaj obliczony tangens. Atan zwraca funkcję odwrotną do funkcji tangens danej liczby.

Kąt = ATAN(liczba)

gdzie **liczba** jest tangensem danego kąta.

Arcus tangens to kąt, którego tangens jest liczbą. Zwrócony kąt jest wyrażany w radianach w zakresie od $-\pi/2$ do $\pi/2$.

Dlatego pamiętaj aby przeliczyć radiany na stopnie, a to robiłeś już w zadaniu 6.

Testuj program wykonując po jednym punkcie.

Sprawdź jaki plik nagłówkowy należy dodać aby działało **sqrt** oraz **atan**!

Dokonaj sprawdzenia poprawności działania programu poprzez wprowadzenie danych: dla $a=15\text{ cm} \rightarrow \text{kąt} = 30$

Widok prawidłowego rozwiązania zadania:

```

C:\Users\andrzej_r\Desktop\Pulpit\Cwiczenia_C++\Zadanie_10_C++\Zadanie_10.exe
Aktywna strona kodowa: 1250
podaj bok a trojkata: 15
Bok trójkąta b = 25.981 cm
Pole trójkąta = 194.856 cm^2
cosinus alfa = 0.866
sinus alfa = 0.500
tangens alfa = 0.577
kąt alfa = 30.000 stopni
Aby kontynuować, naciśnij dowolny klawisz . . .
  
```

ZADANIE 11.

Nazwy zmiennych tak, aby zawierały trzy pierwsze litery nazwiska.

Wykonaj czynności oraz obliczenia

- Wczytać punkty A B C D.
- Wypisz te punkty po wczytaniu np. A(2,-3).
- Obliczyć współrzędne wektorów AB i CD
- Wypisz te wektory po wczytaniu np. AB=[2,-3].
- Oblicz kąt między wektorami AB, CD. Podaj kąt w stopniach w formatach dogodnych dla użytkownika np. do jednego miejsca po przecinku.
- Użyj komentarzy. Zmień kolor tła tego okna oraz kolor pisania.

Wskazówka:

Dla dwóch punktów $A(x_a, y_a)$ i $B(x_b, y_b)$ obliczony wektor AB to:

$$AB = (x_b - x_a, y_b - y_a)$$

np.

Oblicz współrzędne wektora o początku w punkcie $A = (5, 2)$ i końcu w punkcie $B = (-3, 8)$. Oblicz długość wektora \vec{AB} .

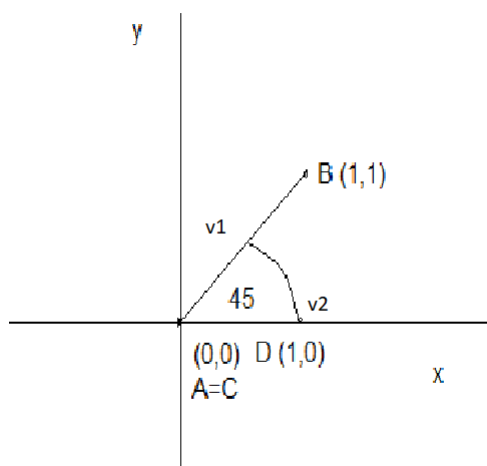
Rozwiązanie:

Najpierw liczymy współrzędne wektora:

$$\vec{AB} = [-3 - 5, 8 - 2] = [-8, 6]$$

Kąt między wektorami obliczmy:

$$v1=[a,b] \quad v2=[c,d]$$



$dv1, dv2$ – długości wektorów. Długość wektora jest to pierwiastek z sumy kwadratów ich współrzędnych, czyli:

$$dv1 = \sqrt{a^2 + b^2}$$

$$dv2 = \sqrt{c^2 + d^2}$$

Wzór na kąt pomiędzy wektorami $v1$ i $v2$ wyprowadzamy ze wzoru na iloczyn skalarny:

$$\cos_alfa := (a*c + b*d) / (dv1 * dv2)$$

Następnie \sin_alfa z jedynki trygonometrycznej potem tangens.

$$\sin^2 \alpha + \cos^2 \alpha = 1$$

$$\sin_alfa = \sqrt{1 - (\cos_alfa)^2}$$

$$\tan_alfa = \frac{\sin_alfa}{\cos_alfa}$$

Do obliczenia kąta użyj $\text{atan}(\text{tangens})$.

$$alfa_x = \text{atan}(\tan_alfa)$$

Z radianów zamień na stopnie.

$$alfa_s = \frac{alfa_x * 180}{\Pi}$$

Dokonaj sprawdzenia poprawności działania programu poprzez wprowadzenie danych zgodnych z rysunkiem poniżej.

Widok prawidłowego rozwiązania zadania:

```

C:\Users\andrzej_r\Desktop\Pulpit\Cwiczenia_C++\Zadanie_11_C++\Zadanie_11.exe
podaj współrzędną y punktu D :
0
A(0.00;0.00)
B(1.00;1.00)
C(0.00;0.00)
D(1.00;0.00)
współrzędne wektora AB wynoszą: [1.00;1.00]
współrzędne wektora CD wynoszą: [1.00;0.00]
dv1= 1.414
dv2= 1.000
cos_alfa= 0.707
sin_alfa= 0.707
tan_alfa= 1.000
alfa_x= 0.785
Kąt między wektorami AB oraz CD wynosi: 45.0 stopni
Aby kontynuować, naciśnij dowolny klawisz . . .

```

ZADANIE 12 sprawdzające wiadomości:

Zenek kupił w warzywniaku jabłuszka i gruszki. Podaj ile zapłacił za jabłka ile za gruszki a ile razem. Zeniu lubi żarciki, więc wagę jabłek podał w gramach 800 a gruszek w tonach 0,0003. Sklepowa zdębiała, ponieważ ceny które znała wyliczone były w kilogramach, ale dała radę bo skończyła XIII LO w Gdańsku i napisała krótki programik w C++. Program napisany został w sposób umożliwiający wprowadzanie różnych wartości wejściowych. Wynik podała wg poniższego wzoru:

$\text{wartość_jabłek} = 800 * 2,25 \text{ zł} = \dots \text{ zł}$
 $\text{wartość_gruszek} = 0.0003 * 4,28 = \dots \text{ zł}$
 $\text{suma} = \dots \text{ zł}$

przypilnuj aby wyświetlane znaki = znalazły się jeden pod drugim

```

C:\Users\andrzej_r\Desktop\Pulpit\Cwiczenia_C++\Zadanie_12_C++\Zadanie_12.exe
Podaj wagę jabłek w gramach:
800
Podaj wagę gruszek w tonach:
0.0003
Podaj cenę jabłek w kilogramach:
2.25
Podaj cenę gruszek w kilogramach:
4.28
wartość jabłek =      0.80 kg *2.25 zł      =      1.80 zł
wartość gruszek =      0.30 kg *4.28 zł      =      1.28 zł
                        suma                =      3.08 zł
Aby kontynuować, naciśnij dowolny klawisz . . .

```

Podziały typów zmiennych:

- typy na fundamentalne i pochodne.
- typy wbudowane (czyli takie, w które C++ jest na starcie wyposażony) oraz definiowane przez użytkownika, czyli takie, które sobie sami wymyślimy.

Typy pochodne

Typami pochodnymi są np. tablica, wskaźnik, funkcja czy referencja.

Typy fundamentalne

Oto one:

Typy całkowite:

short int (lub short)

long int (lub long)

int

enum (typ wyliczeniowy)

Typy rzeczywiste:

float

double

long double

Trzy typy rzeczywiste są po to, żeby programista mógł wybrać dokładność swoich obliczeń. Najmniej w pamięci zajmuje typ float, ale jest przez to najmniej dokładny. Long double zajmuje najwięcej pamięci, ale jest najdokładniejszy.

Typ znakowy: char.

Uwaga:

Typy całkowite i char mogą występować w dwóch odmianach: z lub bez słowa unsigned. Postawienie unsigned przed np int oznacza, że typ ten może reprezentować liczby tylko dodatnie (bez znaku, czyli unsigned). Natomiast w wypadku typu char to jak zostanie zrozumiane słowo unsigned (lub jego brak), zależy od implementacji (czyli od kompilatora). Do zmiennej typu char możemy podstawić dowolną literę (bądź cyfrę) ujętą w apostrofy, np:

```
char znakowa = 'a' ;
```

```
znakowa = '5' ; // przypisanie cyfry, a nie liczby
```

Do zmiennej znakowej można też przypisać jeden z tzw. znaków sterowanych. Jeśli do zmiennej znakowej przypiszemy np. '\n', to zmienna ta będzie zawierała znak nowej linii.

Oto lista wszystkich znaków sterowanych

(mogą się przydać podczas rozwiązywania zadań):

'\b' – cofnięcie o jedną pozycję
 '\f' – nowa strona
 '\n' – nowa linia
 '\r' – powrót karetki
 '\t' – tabulator poziomy
 '\v' – tabulator pionowy
 '\a' – sygnał dźwiękowy
 '\\' – oznaczenie backslasha
 '\'' – apostrof
 '\"' – cudzysłów
 '\0' – NULL, znak o kodzie zero
 '\?' – znak zapytania

Ostatnie pięć znaków służy do wypisywania specjalnych znaków, które są już używane w C++. Przykładowo, żeby wypisać cudzysłów, nie możemy napisać po prostu “, trzeba napisać \“.

Do zmiennych znakowych można też przypisać wartość danego znaku z tablicy ASCII, ale musi to być liczba podana w zapisie ósemkowym bądź szesnastkowym.

String

String ciąg znaków ujęty w cudzysłów, np. “Taki sobie napis”. Trzeba wiedzieć, że komputer przechowuje taki string jako tablicę zmiennych znakowych, czyli w tym przypadku tablica będzie się składać ze zmiennych, które będą miały wartość odpowiednio: T, a, k, itp.

Podczas wprowadzania tekstu do zmiennych łańcuchowych, należy pamiętać o tym by zawsze podawać w specyfikacji formatu odczytywanych danych maksymalna długość tekstu – funkcja scanf() nie jest w stanie sama tego sprawdzić, a pominiecie weryfikacji spowodowałoby, że złośliwy użytkownik mógłby wprowadzić zbyt długi tekst i doprowadzić do zalamania się programu.

```
Char Tekst[128];
```

```
Scanf(„%128s”, &Tekst);
```

Zmienne łańcuchowe są wyjątkowe pod jednym względem – nie mają jednej, wspólnej wartości (w końcu składają się z łańcucha znaków), dlatego kompilator, napotykając sama nazwę zmiennej, zawsze przyjmuje, że autorowi chodziło o adres początku obszaru pamięci zajmowanego przez tekst. Z tego powodu poniższy fragment programu również jest poprawny – mimo braku znaku &:

```
Char Tekst[128];
```

```
Scanf(„%128s”, Tekst);
```

Funkcja scanf() ma jednak poważny brak, wynikający z przystosowania jej do odczytywania bardziej złożonych ciągów danych w jednym przebiegu – nie jest w stanie odczytać tekstu zawierającego odstępy (spacje). Wprowadzenie kilku wyrazów oddzielonych spacjami spowoduje, że do zmiennej łańcuchowej zostanie wpisany tylko pierwszy z nich, a pozostałe zostaną zignorowane.

Typ void,

który oznacza typ: pusty.

Typ enum.

Typ wyliczeniowy. Zmienna typu wyliczeniowego może przyjmować tylko takie wartości, jakie wyliczyliśmy przy definicji. Np.

```
enum dzien {pon, wto, sro, czw=20, pia, sob, nie} ;
 dzien zmienna = pon ;
 zmienna = 21 ; // nielegalne!
```

Trzeba zaznaczyć, że każdy element w liście wyliczeniowej ma jakąś wartość liczbową. W naszym przykładzie: pon=0, wto=1, sro=2, czw=20, pia=21, sob=22, nie=23. Widać, że numeracja zaczyna się od zera (jeśli nie określiliśmy inaczej), natomiast jeśli jakiemuś elementowi nadamy inną wartość (niewynikającą z kolejności), to następne elementy listy będą numerowane zaczynając od tej zmienionej wartości.

Przykład 9

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

using namespace std;

int main(int argc, char *argv[])
{
    system("chcp 1250");
    system("cls");
    char imie[64];
    int wiek;

    printf("Jak masz na imię?: ");
    scanf("%64s", imie);

    printf("Ile masz lat?: ");
    scanf("%i", &wiek);

    printf("\nWitaj %s! Ja mam %i lat!\n", imie, wiek+1);

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

getch() - jest to wywołanie funkcji, która oczekuje na podanie z klawiatury dowolnego znaku. Później wykonywane są dalsze polecenia. Dzięki temu zabiegowi, nasz napis będzie widoczny aż do naciśnięcia jakiegokolwiek przycisku na klawiaturze.

W czasach języka C łańcuchy tekstu tworzyło się za pomocą char'ów i programista musiał wprowadzić maksymalną długość tekstu. W języku C++ wprowadzono udogodnienie jaką jest klasa **string**. Jest ona klasą, która dynamicznie zarządza danymi i alokuje bądź zwalnia sobie pamięć w zależności od potrzeb. Zmienną typu string używa, deklaruje i inicjuje się w następujący sposób:

String nazwa_zmiennej = „w cudzysłowach wpisujemy tekst który ma być w zmiennej”;

Stringi można łączyć ze sobą lub dodawać tekst za pomocą znaku dodawania '+’.

```
String strZmienna = „przy”;
String strZmienna2 = „kład”;
Cout << strZmienna + strZmienna2;
```

W ten sposób w konsoli wyświetli się tekst „przykład”. Można też łączyć więcej zmiennych i tekstów naraz: Przykład po uprzednim zadeklarowaniu i zapisaniu danych do stringów:
`strCalyTekst = strJakisString + „jakis tekst” + strKolejnyString`

Klasa string ma funkcję która podaje nam długość wyrazu wg wzoru.:

strZmienna.length().

Jeśli mamy zmienną typu string możemy również odwoływać się do poszczególnych znaków z łańcucha. Pierwszy znak (podobnie jak w tablicach) ma indeks 0, drugi indeks 1 itp. Dla stringa o długości n znaków znaki będą zapisane kolejno w przedziale <0;n-1>. Np. pierwszy znak zmiennej typu string będzie `strZmienna[0]` a drugi znak `strZmienna[1]`.

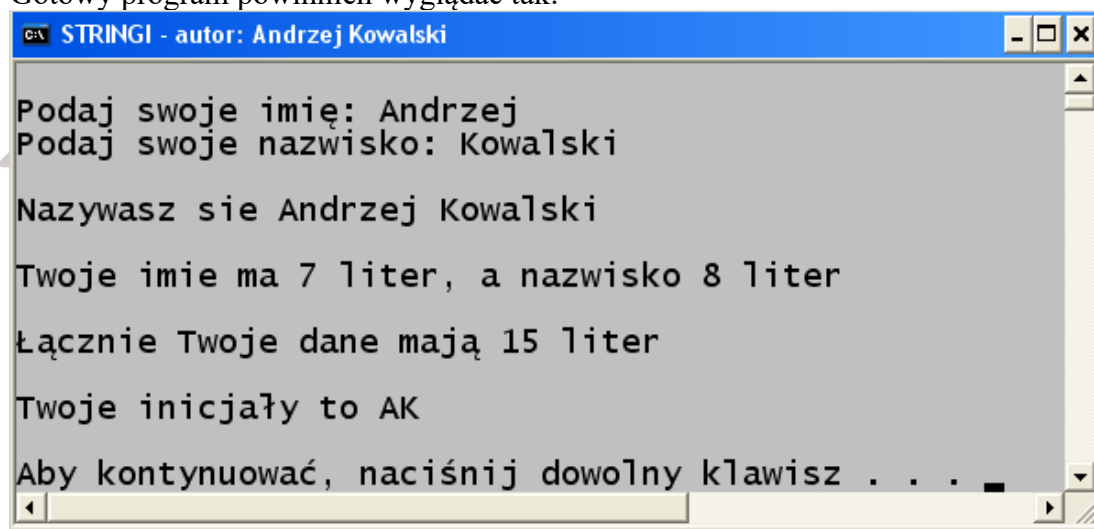
ZADANIE 42.

Napisz program który wczyta twoje imię do zmiennej typu string a potem do kolejnej takiej zmiennej nazwisko. Program ma wypisać długość twojego imienia i nazwiska oraz podać twoje inicjały.

Program powinien zawierać:

- Kolor i tło dokładnie jak na zdjęciu.
- Zmienioną nazwę konsoli. (funkcja systemowa title)
- Wyświetlać polskie znaki, zmieniona strona kodowa nie powinna być widoczna.

Gotowy program powinien wyglądać tak:



ZADANIE 13.

Sporządź wykres prędkości samochodów w m/s. Dane wprowadzaj w km/h. Jako dane wyjściowe podaj wartości prędkości w m/min oraz m/s.

Zbuduj program tak aby przedstawiał wynik dla danej marki samochodu oraz podawał stosowne przeliczenia prędkości maksymalnych, a następnie umieszczał wynik zmierzony w m/s na osi współrzędnych. Podaj miarę osi OX. Ogranicz prędkość samochodów do 280 km/h.

jako liczba typu long. To samo z typem unsigned, tyle że tutaj dopisujemy literę U (bądź małe u).

Zakresy ważności nazw i czas życia obiektów

Czas życia obiektu to okres pomiędzy definicją obiektu (czyli przydzieleniem pamięci dla niego), a destrukcją obiektu (czyli zwolnieniem pamięci). Natomiast zakres ważności nazwy obiektu to fragment (czasami całość) programu, w którym dana nazwa obiektu jest znana kompilatorowi. Jaka jest między nimi różnica?! Ano taka, że w pewnym momencie w programie obiekt może istnieć (czyli żyje, jego czas życia się jeszcze nie skończył), ale może nie być dostępny.

wyróżniamy trzy rodzaje zakresów.

Zakres lokalny to fragment programu objęty parą nawiasów {}.

```
main()
{
    ... // jakieś tam instrukcje
    { // otwarcie bloku lokalnego
        int a; // definiujemy sobie zmienną a
        ... // tutaj nazwa a jest znana, możemy sobie na zmiennej a pracować
    }
    ... // tutaj nazwa a nie jest znana
```

Takim blokiem jest też funkcja, zatem we wnętrzu funkcji zdefiniowane nazwy nie są znane na zewnątrz.

Zakres globalny (pliku) oznacza, że jeśli zdefiniujemy jakąś zmienną na zewnątrz wszystkich funkcji, to jest ona znana w każdej funkcji, czyli: w całym obszarze pliku dana nazwa jest znana. Jeśli chcemy, aby nazwa była znana we wszystkich plikach, czyli w całym programie, to przed deklaracją należy postawić słówko extern, czyli z angielskiego: na zewnątrz.

Zakres obszaru klasy Nazwa użyta wewnątrz definicji klasy jest znana tylko wewnątrz klasy (i dostępna dla jej funkcji składowych)

Przykrywanie zmiennych.

Może się zdarzyć sytuacja, że będzie istniała globalna zmienna o nazwie x, a my w jakimś lokalnym bloku definiujemy drugą też o nazwie x. Taka sytuacja może mieć miejsce (pomimo tego, że są dwie zmienne o tych samych nazwach). W takim razie, skoro funkcja zna wszystkie zmienne globalne, jak ma ona rozróżnić, do której się właśnie zwracamy?? Otóż, jeśli w wyrażeniach będziemy używali x normalnie, tak jakby była jedna zmienna o nazwie x, to funkcja zrozumie, że chodzi nam o zmienną LOKALNĄ. Jeśli natomiast przed nazwą x postawimy dwa dwukropki, czyli będziemy mieć ::x, to funkcja będzie pracowała na zmiennej GLOBALNEJ. Tak się utarło w C++, że operator :: oznacza mniej więcej “przejdźcie na wyższy poziom” (tak przynajmniej ja to odbieram).