Mateusz Wiliński

C++ od podstaw

Podstawy programowania w języku C++

1

--version: 1.0.0

```
cout << "Wszechstronność i szybkość - C++";
```

WILNOX

C++ od podstaw

Podstawy programowania w języku C++

Ebook może być dowolnie rozpowszechniany
i używany w dowolnym celu.

Zapraszamy do współpracy przy jego tworzeniu i ulepszaniu.

Link do aktualnej wersji:

C++ od podstaw

Autor: Mateusz Wiliński
Email: info.wilnox@gmail.com
Data ostatniej aktualizacji: 06.2022

Zapraszamy na stronę https://wilnox.com/

Odwiedź stronę: https://edabit.com/challenges , gdzie znajdziesz mnóstwo zadań związanych z programowaniem w python.

Spis zagadnień

Spis zagadnień 2

Wprowadzenie 6

- Przygotowanie środowiska 6

Podstawowy program 8

- Biblioteki (headery) 8
- Przestrzenie nazw 9
- Funkcja int main(){ } 9
- Zmienne 9
- Wyświetlanie tekstu w konsoli - cout 10
- Wyświetlanie polskich znaków w konsoli 10
- Return 11
- Wstrzymywanie konsoli 11

Zmienne 12

- Jak utworzyć zmienną? 12
- Zasady nadawania nazw 13

Typy danych 14

- Liczby całkowite – int 14
- Liczby rzeczywiste - float, double 14
- Znaki – char 15
- Wartości tekstowe - string 16
- Wartości logiczne - bool 16
- Zasięg zmiennych 18
- Escape codes i stałe 19
- Stałe 19

Wprowadzanie danych do programu 21

Instrukcja warunkowa 23

Operatory porównania 24
Przykład - Obliczanie obwodu koła 25
Operator tenarny ? : 26
Losowa liczba 27
Inkrementacja i dekrementacja 29
Funkcje matematyczne 31
Przykład - Pierwiastek podanej liczby 32
Przykład - pobieranie pieniędzy z z konta 32
Operatory logiczne 33
Przykład - Autoryzacja logowania 34
Przykład - Działanie matematyczne 35
Priorytet operatorów 36
Pętla for 37
Pętla while 38
Switch 40
Funkcja 42
Definicja i wywołanie funkcji 42
Funkcje z parametrem 44
Deklaracja funkcji 46
Przeciążanie funkcji 47
Tablica 47
Wyznaczanie długości tablicy 50
Sortowanie tablicy 52
Zmiana typów 53
Wartości maksymalne danego typu 55
Enum 55

Struct 57

Klasa 59

Wskaźniki 63

Sekcja z zadaniami 66

- Pole trójkąta 66
- Przeliczanie calów na centymetry 67
- Losowa liczba z przedziału 68
- Dodawanie liczb 69
- Dodawanie liczb pobranych z konsoli 70
- Generowanie losowej litery 71
- Generowanie losowego pola(np. a2, b6, itp.) 72
- Wyświetlanie znaków z tekstu 73
- Wypisz liczby: 0, 10, 20, ..., 100 74
- Wypisz liczby: 10,9,8,...,0 75
- Wyświetl tabliczkę mnożenia 76
- Wyświetl losową osobę z tablicy 77
- Losowe działanie z wprowadzaniem odpowiedzi 78
- Podaj promień koła, wyświetl obwód i pole koła (zmienne lokalne) 79
- Wyświetl liczby od 14 do 33, poza liczbą 20 i 30 80
- Przypisz do tablicy 10 losowych liczb i wyświetl rosnąco 81
- Pobierz 3 liczby i wyświetl największą za pomocą tablicy 82
- Podaj wymiary akwarium, wyświetl kubaturę 83
- Zabawne połączenie zdań z dwóch tablic 85
- Zabawne połączenie zdań z dwóch tablic 86
- 5 działań matematycznych z oceną (zmienne lokalne) 88
- Książka 89

Elementy języka C 92

- Wyprowadzanie danych w C – printf() 92

Wczytywanie danych w C - getchar(), scanf() 93
getchar() 93
scanf() 94
Zadania C / C++ 95
Obliczanie drogi i prędkości w ruchu jednostajnie przyśpieszonym 95
Kody ASCII dziesiętnie i szesnastkowo 96
Liczba dziesiętna w postaci ósemkowej i szesnastkowej 97
Obliczanie pola trójkąta prostokątnego 98
Rezystancja dwóch rezystorów połączonych równolegle 99
Wyznacz wartość maksymalną z 3 podanych liczb 100
Wyświetl dzień tygodnia na podstawie liczby 101
Sprawdzić, czy liczba całkowita X jest jednocześnie podzielna przez 7 oraz 3. 102
Obliczanie pierwiastków równania kwadratowego 103
Punkt na płaszczyźnie 104
Liczba dni wybranego miesiąca 105
Pora roku na podstawie dnia i miesiąca 107
Program do obliczania pola figur z menu 108
Policzyć ile wśród 10 liczb wprowadzonych z klawiatury jest dodatnich, ujemnych, zerowych 109
Wczytać 10 liczb całkowitych. Obliczyć sumę liczb dodatnich i ujemnych. 110
Obliczyć i wyświetlić jaką drogę pokona ciało spadające swobodnie po 1, 2, ..,10 sek. 111
Ile razy we wczytanym ciągu N liczb wystąpiła wartość z przedziału od 5 do 25 (włącznie) 112
Obliczyć i wyświetlić kwadraty i sześciany N kolejnych liczb całkowitych 113
Wczytaj ciąg liczb całkowitych zakończony zerem. Policz i wyświet ilość liczb dodatnich i ujemnych w ciągu 114
Sprawdź dla jakiego N suma szeregu 12+22+32+....+N2< 10000 115
Znajdź największą liczbę rzeczywistą, której trzecia potęga jest mniejsza od 2000. 116

Znaleźć NWD (największy wspólny dzielnik) dwóch liczb całkowitych 117
NWD(największy wspólny dzielnik) i NWW (najmniejsza wspólna wielokrotność) 118
NWD(największy wspólny dzielnik) i NWW (najmniejsza wspólna wielokrotność) bez wskaźników 119
W tablicy n elementowej liczb całkowitych policz ilość elementów większych niż średnia wartość tych elementów. 120
Z n elementowej tablicy A do tablicy B skopiuj elementy większe od parametru k. 121
W n elementowej tablicy losowych liczb całkowitych zamień wartości ujemne na ich kwadraty. 122
Wyświetl indeksy i wartości elementów dodatnich tablicy. 123
Liczba elementów tablicy o wartościach z zakresu <p;q> 124
Punkty na płaszczyźnie wewnątrz okręgu 125
Suma każdego wiersza i kolumny w macierzy 6 x 5 126
Numer wiersza o największej sumie elementów 127
Usuwanie wskazanego wiersza z tablicy dwuwymiarowej 128
Wyznaczanie liczby pi za pomocą losowych punktów wewnątrz okręgu 129
Suma cyfr podanej liczby 130
Wypisanie ciągu Fibonacciego do n-tego elementu 131
Struktura, książki 132
Struktura, książki z cin 133
Struktura, książki z cin 134

Wprowadzenie

Przygotowanie środowiska

Zanim zaczniemy uczyć się programować w języku C++, musimy znaleźć odpowiednie środowisko, które Nam to umożliwi. Środowisko to musi rozumieć komendy i instrukcje, które będziemy wpisywać, a następnie tłumaczyć je na język zrozumiały dla komputera, tzw. maszynowy. Program, który tłumaczy kod napisany w języku C++ (dotyczy to również innych języków programowania) nazywamy kompilatorem.
Dodatkowo, poza samą opcja tłumaczenia, możemy również wybrać program (środowisko) oferujące Nam o wiele więcej przydatnych opcji, np. pokazywanie błędów, wyświetlanie podpowiedzi, kolorowanie kodu, dzięki czemu pisanie kodu jest szybsze i łatwiejsze. Takie zintegrowane środowisko nazywamy IDE, integrated development environment. W ramach tego kursu, wypróbujemy kilka różnych darmowych środowisk, w tym również online, ponieważ elastyczność w nauce programowania jest bardzo ważna.

Zacznijmy od wpisania w przeglądarce frazy: free c++ ide
W wynikach wyszukiwania możemy zobaczyć dziesiątki edytorów / IDE, które możemy wykorzystać. Jedne są bardzo dobre, ale wymagają też silniejszego komputera (np. Visual Studio), inne są bardziej ubogie w dostępne funkcje, ale będą działać na starych komputerach. Czy wybór odpowiedniego IDE jest aż tak istotny? Na początku, oczywiście nie, dlatego będziemy eksperymentować z ich wyborem, ale docelowo warto wybrać jeden i poznać go bardziej

szczegółowo. Pamiętajcie, że lepiej znać 1 – 2 narzędzia na wysokim poziomie, niż dziesiątki programów po trochę. Podobnie jest z językami, nikt Was nie zatrudni, jeśli będziecie znać 5 języków programowania w stopniu podstawowym, ale jak poznacie 1 w stopniu zaawansowanym to drzwi stają otworem.
Przejrzymy 3 środowiska, takie jak: DEV C++, Visual Studio oraz kompilator online - jdoodle, dzięki któremu możemy pisać kod bezpośrednio w przeglądarce internetowej.

Praktykuj programowanie na platformie:
https://www.hackerrank.com/domains/cpp

Podstawowy program

```
#include <iostream>
using namespace std;

int main() {
    int x = 10;
    int y = 25;
    int z = x + y;

    cout << "Sum of x+y = " << z;
}
```

Biblioteki (headery)

Biblioteki to porcje kodu, które rozszerzają funkcjonalność i możliwości tworzonego programu. Nowy projekt nie zawiera zbyt wiele możliwości, aby nie znajdowało się w nim zbyt wiele niepotrzebnych linii kodu, które negatywnie wpływają na wydajność i rozmiar programu. Kolejne biblioteki dodajemy wtedy, kiedy zachodzi potrzeba użycia funkcji, które oferują.
Nowe biblioteki dodajemy za pomocą instrukcji #include, dodając nazwę biblioteki wewnątrz nawiasu ostrego, czyli deklaracja `#include <iostream>` dołącza wbudowaną bibliotekę o nazwie iostream. Biblioteka ta obsługuje kanały wejściowe i wyjściowe, dzięki czemu możemy np. wypisać coś w konsoli programu. Większość bibliotek, które będziemy potrzebować są już wpisane w pamięci programu, wystarczy je tylko zadeklarować na początku programu, tak jak jest to pokazane w przykładzie. Czasami będziemy chcieli użyć bibliotek zewnętrznych, które najpierw należy zaimportować do programu, co zostanie omówione w momencie kiedy przyjdzie taka potrzeba.

Biblioteka iostream służy do obsługi strumieni wejściowych i wyjściowych, za jej pomocą możemy wypisywać treści w konsoli lub wprowadzać dane do programu.

```
#include <iostream>
```

Przestrzenie nazw

Używanie przestrzeni nazw pozwalają używać dostępnych opcji w łatwiejszy, szybszy sposób. Dodajemy je za pomocą instrukcji using namespace, np.:

```
using namespace std;
```

Funkcja int main(){ }

Funkcja main to główna funkcja każdego programu, wewnątrz niej zaczyna się i kończy napisany przez Nas program. Jak każda funkcja składa się z typu, nazwy, nawiasu okrągłego i klamrowego.

```
int main() {}
```

Zmienne

Trzy kolejne linie:

```
    int x = 10;
    int y = 25;
    int z = x + y;
```

To deklaracje zmiennych o typie integer, czyli o typie liczb całkowitych. Do podanych nazw x, y, z przypisuje się konkretne wartości. Więcej o zmiennych w dziale Zmienne i ich typy.

Wyświetlanie tekstu w konsoli - cout

Informacje wyświetlane są w oknie konsoli za pomocą obiektu cout, tak jak zostało to pokazane w programie przykładowym:

```
cout << "Sum of x+y = " << z;
```

Po instrukcji cout podaje się obiekt <<, który może łączyć różne typy informacji. W tym przykładzie łączymy wartość tekstową zawartą w cudzysłów z wartością liczbową z.

Wyświetlanie polskich znaków w konsoli

Na początku programu, w funkcji main, dodaj poniższą instrukcję:

```
system("chcp 1250>nul");
//lub
setlocale(LC_CTYPE, "Polish");
```

Return

Słowo return oznacza, że funkcja zwraca określoną wartość. W przypadku funkcji main zazwyczaj wpisujemy instrukcję return 0; lub pomijamy ją, ponieważ kompilator dodaje ją automatycznie w momencie kompilacji.
Instrukcji return używamy zawsze wtedy, gdy funkcja ma typ zwrotny inny niż void.

Wstrzymywanie konsoli

W niektórych kompilatorach należy wstrzymać konsolę by przeczytać wynik programu. W tym celu można zastosowac jedną z poniższych instrukcji, choć w większości przypadków jest to zbędne.

```
system("pause");
//lub
getchar();
```

Zmienne

Zmienna, to wirtualny kontener, który ma za zadanie przechowywać wartości danego typu. W programowaniu wyróżniamy kilka typów informacji, np.: dane liczbowe (całkowite i rzeczywiste), znaki, łańcuchy znaków, wartości logiczne (prawda lub fałsz).

Jak utworzyć zmienną?

Tworzenie zmiennej nazywamy deklaracją, robimy to podając jej typ oraz nazwę.

```
typZmiennej nazwaZmiennej;
```

Później możemy przypisać do utworzonej zmiennej wartość, np.:

```
int liczbaMieszkancow;
liczbaMieszkancow = 120000;
```

Podczas deklaracji nie musimy podawać żadnej wartości, choć oczywiście możemy. Przypisując wartość do zmiennej podczas jej tworzenia nazywamy inicjalizacją, np.:

```
int liczbaMieszkancow = 120000;
```

Następnie możemy zmienić zawartość 'wirtualnego kontenera' przez nadpisanie zmiennej nową wartością
(ale o tym samym typie), np.:

```
liczbaMieszkancow = 50000;
```

Typ danej zmiennej podajemy tylko raz, podczas deklaracji, później wszelkie odwołania do takiej zmiennej są przeprowadzane tylko za pomocą jej nazwy. Stąd bardzo ważny wniosek:
każda zmienna musi mieć unikalną nazwę.

Zasady nadawania nazw

Mówiąc o nazwach, musimy wspomnieć o kilku ważnych zasadach:

- nazwy rozpoczynamy od małej litery, a kolejne słowo składowe zaczynamy z wielkiej litery (jest to tak zwana notacja camelCase, np: numerButa, dataUrodzenia, kolorNadwozia, itp);
- nazwy muszą być opisowe, tak, aby łatwo było domyślić się, co znajduję się w danej zmiennej (przykład złej nazwy to: zmienna2, cos, zm2);
- należy stosować litery alfabetu angielskiego (bez polskich akcentów czy symboli);
- można korzystać z liczb, ale nie można od nich zaczynać nazwy;
- istnieje kilka zarezerwowanych słów kluczowych, których nie można używać jako nazwy zmiennych (przeczytaj o nich w internecie - C++ słowa kluczowe);

Typy danych

Liczby całkowite – int

Zmienna typu int (integer) przechowuje wartości liczb całkowitych. Każdy typ zmiennej zajmuje określony rozmiar w pamięci, np: 4 bajty. Rozmiar ten definiuje zakres możliwych wartości, które można w takiej zmiennej przechować. W przypadku zmiennej typu int, maksymalną liczbą, jest liczba 2147483647

Przykład: int liczbaMieszkancow;

Liczby rzeczywiste - float, double

Bardzo często zachodzi potrzeba posługiwania się liczbami z częścią ułamkową, np: 1.22, 10.5, 36.6, itd.
Należy zapamiętać, że od strony kodu, przecinki służą do odzielania argumentów (o których będziemy mówić później), a do rozdzielania części ułamkowej w liczbie, służy kropka. Napiszemy więc wartość 12.3, a nie 12,3!

Wartości ułamkowe są przechowywane wewnątrz zmiennej typu float i double. Typ double jest 2-krotnie większy od float, jak sama nazwa wskazuje 😊 i posiada 2-krotnie większą precyzję.
Typ float ma precyzję ok 6-9 cyfr, natomiast typ double ok 15-18.

Przykład

```
float temp = 12.5;
double odlegloscOdSlonca = 1863476324793.45;
```

Precyzja liczb zmiennoprzecinkowych oznacza, że np. typ float, który ma precyzję od 6 do 9 cyfr (w zależności od systemu), pokaże tylko tyle tych liczb, a resztę zamieni na 0.

```
float temp = 12.58678769050;
std::cout << temp;
```

Powyższy przykład zwróci wynik: 12.5868, reszta cyfr zostaje odrzucona.

Znaki – char

Chcąc wyświetlić w konsoli litery, cyfry lub inne znaki, jak ! @ itp. należy użyć typu char. Możemy do takiej zmiennej przypisać określony znak z klawiatury z użyciem apostrofu lub podać jego numer z tablicy ASCII. Poniższe zapisy są równoznaczne:

```
char znak = 'A';
char znak2 = 65;
```

Wartości tekstowe - string

Zmienne czy ogólnie wartości typu string to wartości tekstowe. Należy na nie patrzeć jak na tablicę pojedynczych znaków. Aby korzystać ze zmiennych typu string, należy dołączyć header #include <string> choć nie jest to wymagane, gdy korzystamy z headera <iostream>. Typ string działa w przestrzeni std, dlatego należy przed nim wpisać std::string lub dołączyć instrukcję using namespace std;

Zmienne typu string mają kilka charakterystycznych metod, np.: obliczanie ilości znaków w łańcuchu za pomocą metody length() lub wyłonienie części znaków za pomocą metody substr().

```
#include <iostream>
using namespace std;

int main() {
     string miasto = "LEGNICA to miasto z historią";
     cout << miasto.length() << endl;
     cout << miasto.substr(miasto.length()-10,10);
}
```

Wartości logiczne - bool

Dla komputera istnieją 2 stany: jest to albo prawda albo fałsz. Wyrażenia, które ewoluują do jednej z tych dwóch wartości będziemy spotykać w programowaniu bardzo często. Będziemy je nazywać wyrażeniami logicznymi. W kolejnych etapach nauki (ale nie w tym kursie) poznacie prawa boola, które są ściśle związane z operacjami logicznymi. W algebrze boola, jak i również w programowaniu,

prawda oznaczana jest przez cyfrę 1, a fałsz przez cyfrę 0. W tym kursie poznamy podstawy, które w zupełności wystarczą, by napisać wiele użytecznych programów.

```
bool czyZdanieJestPrawdziwe = true;
bool czyZdanieJestPrawdziwe = 1;
//lub
bool czyZdanieJestPrawdziwe = false;
bool czyZdanieJestPrawdziwe = 0;
```

Po przypisaniu wartości do zmiennych możemy je oczywiście wypisać w konsoli, tak jak robiliśmy to wcześniej z innymi typami zmiennych. Niektóre języki wyświetlają wartości w postaci tekstu (true, false) podczas, gdy inne (C++) w postaci numerycznej (1,0).

Wyrażenie logiczne zawsze musi przyjąć jednoznaczną wartość, prawdę lub fałsz, dlatego też, musi być odpowiednio sformułowane.

Przykład 1

```
#include <iostream>
int main() {
    bool czyZdanieJestPrawdziwe = 20 > 2 * 4;
    std::cout << czyZdanieJestPrawdziwe;
}
```

Wyrażenie jest prawdziwe, ponieważ 20 jest większe od 8 (2*4).

Zasięg zmiennych

Zmienne możemy deklarować wewnątrz funkcji, tak jak robiliśmy to do tej pory. Jest to najlepszy i preferowany sposób. Dzięki takiemu zabiegowi zmienna ta "żyje" wewnątrz tej funkcji. Nie można jej zmienić czy odnieść się do niej z zewnątrz. Zmienną taką nazywamy zmienną lokalną, ponieważ egzystuje lokalnie wewnątrz tej funkcji.

Zmienną, którą deklarujemy poza funkcją, nazywamy globalną i jest ona dostępna dla wszystkich funkcji dostępnych w programie. Zmienne o zasięgu globalnym najlepiej umieszczać na samym początku programu, przed zdefiniowaniem jakichkolwiek funkcji, aby być pewnym, że są one gotowe do użycia.

Zmienna globalna liczba

```
#include <iostream>
using namespace std;

int liczba = 10;

int main(){
     cout << liczba;
}
```

Zmienna lokalna liczba

```
#include <iostream>
using namespace std;

int main(){
     int liczba = 10;
     cout << liczba;
}
```

Escape codes i stałe

Wewnątrz cudzysłowu możemy podawać specjalne znaki, tzw. escape codes, które wykonują odpowiednie czynności w programie, np. przenoszą kursor do nowej linii, tworzą tabulację, itp. Możemy je podawać również w systemie szesnastkowym.

\a = \x07 = alert (bell)
\b = \x08 = backspace
\t = \x09 = horizonal tab
\n = \x0A = newline (or line feed)
\v = \x0B = vertical tab
\f = \x0C = form feed
\r = \x0D = carriage return
\" = quotation mark (backslash not required for '"')
\' = apostrophe (backslash not required for "'")
\? = question mark (used to avoid trigraphs)
\\ = backslash

Stałe

Stałe tworzymy podobnie jak zmienne, ale dodajemy przed nimi słowo const. W języku C stosuje się słowo define. Stałych nie można nadpisywać w trakcie trwania programu i zazwyczaj ich nazwę tworzy się z wielkich liter.

```
#include <iostream>
using namespace std;

#define STALA_1 12       //dla C
const int STALA_2 = 24;  //dla C++

int main() {
    cout << " stala1 : " << STALA_1 << endl;
    cout << " stala2 : " << STALA_2 << endl;
}
```

Wprowadzanie danych do programu

Informacje wprowadzamy do systemu za pomocą instrukcji cin. Aby mogła ona działać należy dodać header <iostream> oraz używać przestrzeni nazw std. Zazwyczaj będziemy pobierać dane przypisując je bezpośrednio do wcześniej zadeklarowanej zmiennej.

```
#include <iostream>
using namespace std;

int main() {
    int odp;
    cin >> odp;
}
```

Metoda cin pobiera tylko znaki do pierwszego białego znaku, aby obrać pełne zdanie można użyć metody getline (cin, mystr);
Wtedy należy również dodać header <string>

```
#include <iostream>
#include <string>
using namespace std;

int main() {
     system("chcp 1250>nul");
     string nazwa;
     cout << "Podaj imię i nazwisko: ";
     getline(cin, nazwa);
     cout << endl << "Witaj " << nazwa;
}
```

Istnieje możliwość pobrania wielu wartości za pomocą jednej instrukcji cin, np.:

```
#include <iostream>
using namespace std;

int main() {
     int num1, num2, sum;
     cout << "Podaj 2 liczby";
     cin >> num1 >> num2;
     sum = num1 + num2;
     cout << "Suma = " << sum;
}
```

Instrukcja warunkowa

Podstawowa instrukcja warunkowa (przedstawiona obok) obsługuje 2 warunki.
Jeżeli wyrażenie logiczne będzie prawdziwe, to kompilator zajrzy do środka instrukcji if i wykona znajdujące się tam instrukcje 1. Reszta, czyli blok else, zostanie pominięty.
Jeżeli wyrażenie logiczne będzie fałszywe, to kompilator od razu przejdzie do bloku else i wykona instrukcje w nim zawarte (instrukcje 2).
Kod jest czytany od góry i w takiej kolejności też będą sprawdzane warunki instrukcji warunkowej. W związku z tym, należy odpowiednio tworzyć kolejne bloki if i else if.

Szablon instrukcji warunkowej

```
if ( wyrażenie logiczne)
{
    instrukcja 1;
}
else
{
    instrukcja 2;
}
```

Operatory porównania

Należy zwrócić uwagę, że przyrównując 2 wartości do siebie, używamy operatora ==. Początkujący programiści często próbują wstawić w to miejsce pojedynczy znak =, co skutkuje błędem. Pojedyńczy znak = służy do przypisania wartości, a nie ich porównywania.

>	... jest większe od
>=	... jest większe lub równe niż
<	... jest mniejsze od
<=	... jest mniejsze lub równe niż
==	... jest równe
!=	... jest różne od

Przykład - Obliczanie obwodu koła

Wyświetl obwód koła na podstawie promienia podanego przez użytkownika
Zdefiniuj stałą PI
Pobierz dane od użytkownika
Oblicz obwód koła (2*PI*r)
Upewnij się, że program wykona się dla odpowiedniej wartości danych

Rozwiązanie

```
#include <iostream>
using namespace std;
#define PI 3.1416f

int main() {
     float r;
     cout << "Podaj promien kola: ";
     cin >> r;
     if (r > 0) {
          float obwod = 2 * PI * r;
          cout << "Obwod wynosi: " << obwod;
     }
     else {
          cout << "Podano zla wartosc. Komputer ulegnie
autodestrukcji!";
     }
}
```

Operator tenarny ? :

Operator tenarny zastępuję prostą instrukcje warunkową i działa w ten sposób, że na podstawie określonego warunku może przypisać do zmiennej odpowiednią wartość.

```
string odp = 10 < 20 ? "OK" : "NOK";

bool odpPoprawna = 5.45 < 4 ? true : false;
```

Losowa liczba

W celu wygenerowania losowej liczby korzystamy z tzw. ziarna, czyli funkcji, która zmienia wynik generowanej liczby: srand();

Funkcja ta przyjmuje argument liczbowy, ale by ta zmienność była kontynuowana w czasie, w miejsce argumentu wstawia się funkcję, która zwraca ilość sekund, które minęły od roku 1970.

```
time_t sekundy = time(0);
cout << sekundy;
```

W celu stworzenia ziarna, łączymy obie funkcje:

```
srand(time(0));
```

Dzięki temu za każdym uruchomieniem programu (po każdej sekundzie) otrzymujemy inne wyniki. Należy pamiętać, że ta metoda jest odpowiednia do prostych czynności, ale nie stosuje się jej w poważnych zastosowaniach.

Wymagane headery:

#include <cstdlib>

```
#include <ctime>
```

Funkcja generująca liczbę to: rand(). Poprzez jej odpowiednie przekształcenie z użyciem operatora modulo, można otrzymać wartość losową z odpowiedniego przedziału, który można zdefiniować poniższym wzorem:

rand() % (max + 1 – min) + min

Przykład

Wyświetl losową liczbę z zakresu od 10 do 20

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
     srand(time(0));
     int min = 10, max = 20;
     int losowaLiczba = rand() % (max + 1 - min) + min;
     cout << losowaLiczba;
}
```

W przypadku chęci wylosowania wartości od 0 do 1 możemy wylosować liczby od 0 do 100, a następnie podzielić ją przez 100, np.:

```
#include <iostream>
#include <ctime>

int main() {
     srand(time(0));
     float losowaLiczba = rand() % 100 / 100.0;
     std::cout << losowaLiczba;
}
```

Inkrementacja i dekrementacja

Inkrementacja i dekrementacja to zwiększanie i zmniejszanie wartości zmiennych o pewien współczynnik. Możemy tutaj używać wielu operatorów arytmetycznych, takich jak:
+, -, * , /, %

Chcąc zwiększyć zmienną o 1 możemy do takiej zmiennej przypisać ją samą powiększoną o 1.

```
int licznik = 0;
licznik = licznik + 1;
```

Powyższy zapis możemy skrócić, poprzedzając operator przypisnaia (=) odpowiednim operatorem arytmetycznym:

```
int licznik = 0;
licznik += 1;
```

W przypadku, gdy powiększamy (lub pomniejszamy) o 1 możemy użyć jeszcze krótszej formy:

```
int licznik = 0;
licznik++;  //post inkrementacja
licznik--;  //post dekrementacje
//lub
++licznik;  //pre inkrementacja
--licznik;  //pre dekrementacja
```

Inne przykłady inkrementacji i dekrementacji:

```
int licznik = 0;
licznik += 2;
licznik *= 3;
licznik /= 2;
licznik = licznik - 1;
licznik %= 2;
cout << licznik;
```

Funkcje matematyczne

W celu stosowania metod matematycznych, należy dodać header:

```
#include <cmath>        //C++
#include <math.h>  //C
```

Podstawowe funkcje:

sqrt(x)	pierwiastek z liczby x
pow(x,y)	liczba x do potęgi y
pow(x,1/y)	pierwiastek liczby x stopnia y
round(x)	zaokrągla liczbę x
ceil(x)	zaokrągla liczbę x w górę
floor(x)	zaokrągla liczbę x w dół
abs(x)	wartość bezwzględna liczby x
fabs(x)	wartość bezwzględna liczby ułamkowej x
modf(x,&y)	rozdziela liczbę rzeczywistą, zwraca część ułamkową z liczby x i przypisuje część całkowitą do zmiennej y

M_PI	zwraca wartość liczby pi
M_E	zwraca wartość liczby Eulera e

Przykład - Pierwiastek podanej liczby

Wprowadź liczbę, a następnie wyświetl jej pierwiastek

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
	system("chcp 1250>nul");
	float liczba;
	cout << "Obliczanie pierwiastka. Podaj liczbę: ";
	cin >> liczba;
	cout << endl << "Pierwiastek liczby " << liczba << "
wynosi: " << sqrt(liczba);
}
```

Przykład - pobieranie pieniędzy z z konta

```
#include <iostream>
using namespace std;

int main() {
	system("chcp 1250>nul");

	double stanKonta = 1000;
	double kwotaDoWybrania;
	cout << "Kwota do wybrania: ";
	cin >> kwotaDoWybrania;
	bool czyMoznaWybrac = kwotaDoWybrania <= stanKonta;

	if (czyMoznaWybrac) {
		stanKonta -= kwotaDoWybrania;
		cout << "Wybrano: " << kwotaDoWybrania << endl;
		cout << "Pozostało: " << stanKonta;
	}
	else {
		cout << "Za mało środków na koncie. Stan konta:" <<
stanKonta;
	}
}
```

Operatory logiczne

Koniunkcja &&

(Wyrażenie jest prawdziwe, gdy wszystkie składowe wyrażenia również są prawdziwe)

```
bool czyZdanieJestPrawdziwe = 13 > 10 && 100 > 99;
std::cout << czyZdanieJestPrawdziwe << std::endl;
```

Alternatywa ||

(Wyrażenie jest prawdziwe, gdy choć jedno wyrażenie składowe jest prawdziwe)

```
bool czyZdanieJestPrawdziwe = 13 > 10 && 100 > 99;
std::cout << czyZdanieJestPrawdziwe << std::endl;
```

Negacja !

(Odwraca wartość całego wyrażenia)

```
bool czyZdanieJestPrawdziwe = !(13 > 10);
std::cout << czyZdanieJestPrawdziwe << std::endl;
```

Przykład - Autoryzacja logowania

```
#include <iostream>
using namespace std;

int main() {
	system("chcp 1250>nul");
	string haslo = "1234", login = "abc";
	string hasloAdmin = "4321", loginAdmin = "cba";
	string hasloPodane, loginPodany;
	cout << "Podaj login: ";
	cin >> loginPodany;
	cout << "Podaj haslo: ";
	cin >> hasloPodane;
	bool daneUzytkownikaZgodne = login == loginPodany && haslo
== hasloPodane;
	bool daneAdminaZgodne = loginAdmin == loginPodany &&
hasloAdmin == hasloPodane;

	if (daneUzytkownikaZgodne || daneAdminaZgodne) {
		cout << "Zalogowano";
	}
	else {
		cout << "Błąd!";
	}
}
```

Przykład - Działanie matematyczne

1.Napisz program, który za każdym razem wyświetli działanie matematyczne (dodawanie) z losowymi składnikami z zakresu (5,20), np.: Oblicz: 7 + 14 =
2.Rozszerz punkt 1 tak, aby można było dodać odpowiedź, a po jej udzieleniu program wyświetli komunikat czy odpowiedź była poprawna czy też nie, użyj operatora tenarnego.

Rozwiązanie

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
     system("chcp 1250>nul");
     srand(time(0));
     int min = 5, max = 20;
     int wynik, odp;
     int l1 = rand() % (max + 1 - min) + min;
     int l2 = rand() % (max + 1 - min) + min;
     cout << "Oblicz: " << l1 << " + " << l2 << " = ";
     wynik = l1 + l2;
     cin >> odp;
     string komunikat = wynik == odp ? "Brawo!" : "Spróbuj
ponownie";
     cout << komunikat;
}
```

Priorytet operatorów

	Operator	Opis	Kierunek
1	::	zakres, przestrzeń nazw	>>>
2	a++ a-- type() type{} a() a[ ] . ->	inkrementacja przyrostkowa rzutowanie funkcjonalne wywołanie funkcji dostęp do obiektów	
3	++a --a ! ~ (type) &a *a sizeof new delete	inkrementacja przedrostkowa zaprzeczenie logiczne i binarne rzutowanie adres dynamiczne lokowanie pamięci (tworzenie, usuwanie obiektów)	<<<
4	.* -> *	wskaźnik	>>>
5	* / %	mnożenie, dzielenie, modulo	
6	+ -	dodawanie, odejmowanie	
7	<< >>	przesunięcie bitów	
8	< > <= >=	porównywanie	
9	== !=	porównywanie	
10	&	iloczyn bitowy	
11	^	różnica symetryczna XOR	
12	|	suma bitowa	
13	&&	iloczyn logiczny	
14	||	suma logiczna	
15	x ? y : z	operator warunkowy(tenarny)	<<<
16	= *= /= %= += -= >>= <<= &= ^= !=	przypisania przypisania binarne	
17	throw	throw expression	
18	,	operator przecinkowy do grupowania wyrażeń	>>>

Pętla for

Za pomocą pętli można wykonywać powtarzające się instrukcje.

Syntax pętli for

```
for (inicjalizacja_i; warunek; modyfikator) {
    // instrukcje do wykonania w każdej iteracji pętli
}
```

Przykład

Wypisz 5-cio krotnie wyraz "Hello" za pomoca pętli for.

```
#include <iostream>
int main(){
    for (int i = 1; i <= 5; i++) {
        std::cout << "Hello" << std::endl;
    }
}
```

Przykład

Wypisz w kolumnie liczby od 1 do 10 za pomoca pętli for.

```
#include <iostream>
int main(){
    for (int i = 1; i <= 10; i++) {
        std::cout << i << std::endl;
    }
}
```

Przykład

Wypisz w kolumnie liczby parzyste od 0 do 20 za pomoca pętli for.

```
#include <iostream>
int main() {
     for (int i = 0; i <=20; i+=2){
          std::cout << i << std::endl;
     }
}
```

Pętla while

Pętle while można, w większości przypadków, stosować zamiennie z pętlą for. Pętla while sprawdzi się w przypadku, gdy nie wiemy ile dokładnie razy ma się ona wykonać.

Syntax pętli while

```
int i = 0;    // inicjalizacja zmiennej iteracyjnej;
while (warunek_wykonania_pętli) {
     // instrukcje do wykonania w każdej iteracji pętli
     // modyfikator iteracji
}
```

Należy pamiętać o zawarciu wyrażenia modyfikującego zmienną 'i'. Każda pętla musi posiadać warunek kończący jej działanie. Pętla, która nie będzie miała takiego punktu, może skutkować przepełnieniem pamięci i zawieszeniem systemu.

Przykład

Wypisz w kolumnie liczby od 1 do 10 za pomoca pętli while.

```
#include <iostream>
int main(){
     int i = 1;
     while (i <= 10) {
          std::cout << i << std::endl;
          i++;
     }
}
```

Przykład

Program, który wyświetla zapytanie o hasło, dopóki nie zostanie udzielona poprawna odpowiedź.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
     system("chcp 1250>nul");
     string haslo = "tajne";
     string odp;
     while (haslo != odp) {
          cout << "Podaj hasło: ";
          cin >> odp;
     }
     cout << "Hasło poprawne!";
}
```

Switch

Instrukcja switch działa podobnie do instrukcji warunkowej if. Za jej pomocą możemy sterować programem i dokonywać odpowiednich decyzji. Wyobraźmy sobie program, który za pomocą wprowadzonych liczb, uruchamia odpowiednie funkcje.
Bardzo ważne jest zawarcie instrukcji break, w przeciwnym razie po wykonaniu danej funkcji program przejdzie do kolejnego kroku, co w tym przypadku nie byłoby wskazane.
Dzięki zastosowaniu instrukcji default, obsłużymy sytuację, kiedy użytkownik wpisze coś niezgodnego z instrukcją. Poniższe przykłady pokazują switcha, który działa na podstawie wartości calkowitych, ale równie dobrze, zamiast wartości 1,2,3 możemy wstawić znaki char czy wartości ułamkowe.

Syntax switch

```
int opcjaWyboru;
std::cin >> opcjaWyboru;

switch (opcjaWyboru) {
case 1: //instrukcja 1
     break;
case 2: //instrukcja 2
     break;
default: //instrukcja defaultowa
     break;
}
```

Przykład: Program z wyborem opcji

```
#include <iostream>
using namespace std;

int main() {
     system("chcp 1250>nul");
     cout << "Wpisz odpowiednią opcję:" << endl;
     cout << "1: Program do obliczania przeciwprostokątnej
trójkąta" << endl;
     cout << "2: Obliczanie rezystancji zast. R1 i R2
połączonych równolegle" << endl;
     cout << "3: Program do obliczania objętości bryły o
wymiarach x,y,z" << endl;

     int opcjaWyboru;
     cin >> opcjaWyboru;

     switch (opcjaWyboru) {
     case 1: cout << "Wybrano program 1";
          break;
     case 2: cout << "Wybrano program 2";
          break;
     case 3: cout << "Wybrano program 3";
          break;
     default: cout << "Nie wybrano odpowiedniej opcji.";
          break;
     }
}
```

Funkcja

Definicja i wywołanie funkcji

Do tej pory wszystkie instrukcje tworzyliśmy wewnątrz funkcji main. Wraz z postępem, jaki czynimy, musimy zacząć rozbijać nasz kod na mniejsze bloki. Funkcja, z założenia, powinna wykonywać prostą operację, co zwiększa czytelność kodu i znacznie pomaga w jego refaktoryzacji(poprawie).
Funkcja składa się z typu zwracanej wartości (podobnie jak zmienne), z tą różnicą, że dla funkcji istnieje dodatkowy typ : void, który oznacza, że funkcja nie zwraca żadnej wartości.
Każda funkcja, która zwraca wartość, np: int, string, double, musi zawierać instrukcję return, która precyzuje, co dana funkcja zwraca. W przypadku głównej funkcji main(), nie wpisywaliśmy tej intrukcji, ponieważ kompilator robił to za nas w ukryciu.
Wewnątrz nawiasu okrągłego możemy podać parametry, choć nie zawsze musimy to robić. Słowo parametry zostało napisane wewnątrz nawiasu kwadratowego, co oznacza możliwość, a nie przymus.

Syntax funkcji (definicja funkcji)

```
typ nazwaFunkcji(parametry)
{
     //ciało funkcji;
     [return value;]
}
```

Sama definicja funkcji nie wiążę się z jej wykonaniem. To tak, jakby napisać plan ćwiczeń na kartce, ale fizycznie ich nie wykonywać.

Funkcję należy wywołać, aby ją uruchomić, robimy to podając jej nazwę i nawias okrągły (wewnątrz, którego podajemy argumenty, jeśli definicja funkcji je określa).

```
nazwaFunkcji(argumenty);
```

Przykład
Napisz program, który wypisze powitanie "Dzień doberek" z użyciem funkcji.

```
#include <iostream>

void wyswietlPowitanie(){
     std::cout << "Dzien doberek";
}

int main() {
     wyswietlPowitanie();
}
```

Funkcje z parametrem

W definicji funkcji możemy zadeklarować parametry, które będą przekazywane do wnętrza funkcji w chwili jej wywołania.
Należy zapamiętać, że w definicji funkcji tworzymy parametry, a wywołując funkcję wpisujemy odpowiadające im argumenty.
Załóżmy, że w programie często wypisujemy dane w konsoli. Zamiast ciągle wpisywać std::cout << ....; możemy stworzyć funkcję, do której będziemy przekazywać treść do wyświetlenia bez powtarzania instrukcji wypisującej cout. Ponieważ funkcja ta nie będzie zwracać żadnej wartości, a jedynie coś wyświetlać, jej typ ustawimy jako void.

Przykład

Program wypisujący dane w konsoli z użyciem funkcji z parametrem

```
#include <iostream>
#include <string>

void wyswietlInfo(std::string info){
     std::cout << info << std::endl;
}

int main() {
     wyswietlInfo("To jest pierwsza wiadomosc");
     wyswietlInfo("A to druga");
     wyswietlInfo("Wow!");
}
```

Przykład

Program zwracający obwód koła, dzięki funkcji przyjmującej promień koła jako argument

```
#include <iostream>

float zwrocObwodKola(float r){
     return 2 * 3.14 * r;
}

int main() {
     std::cout << zwrocObwodKola(3);
}
```

Deklaracja funkcji

Pamiętaj, że kompilator analizuje program od góry w dół. Aby użyć danej funkcji wewnątrz funkcji main() kompilator musi ją wcześniej "poznać", zatem musimy ją zdefiniować przed funkcją main(). Innym sposobem, jeśli chcemy tworzone funkcje umieścić pod funkcją main, jest deklaracja funkcji. Deklarując funkcję mówimy kompilatorowi, hej, wspominam Ci o tej funkcji tutaj, ponieważ znajduje się ona daleko na dole, a ja chcę jej użyć. Funkcję deklarujemy następująco:

```
float zwrocObwodKola(float r);
```

Używając deklaracji funkcji możemy teraz spokojnie przenieść całą funkcję poniżej funkcji main.

```
#include <iostream>
float zwrocObwodKola(float r);

int main() {
     std::cout << zwrocObwodKola(3);
}

float zwrocObwodKola(float r) {
     return 2 * 3.14 * r;
}
```

Przeciążanie funkcji

Przeciążanie funkcji stosujemy wtedy, gdy dana funkcja powinna przyjmować różne typy parametrów lub ich różną liczbę. Jako przykład możemy podać program, który wyświetla losową liczbę. Możemy stworzyć 3 funkcje o tej samej nazwie (co nazywamy właśnie przeciążaniem funkcji). Jedna funkcja będzie losować zawsze liczbę od 1 do, druga funkcja będzie posiadała parametr, opsiujący maksymalny zakres losowanej liczby, a trzecia funkcja będzie posiadała 2 parametry, opsiujące zakres losowanych liczb, min i max.

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

int generujLosowaLiczbe() {
     return rand() % 100 + 1;
}

int generujLosowaLiczbe(int max) {
     return rand() % max + 1;
}

int generujLosowaLiczbe(int min, int max) {
     return rand() % (max - min + 1) + min;
}

int main() {
     srand(time(0));
     std::cout << generujLosowaLiczbe() << std::endl;
     std::cout << generujLosowaLiczbe(10) << std::endl;
     std::cout << generujLosowaLiczbe(10,60) << std::endl;
}
```

Tablica

Tablica, to niezwykle przydatny obiekt, dzięki któremu możemy przechowywać wiele wartości tego samego typu. Wyobraźmy sobie sytuację, że chcielibyśmy przechowywać imiona wszystkich pracowników. Moglibyśmy tworzyć dla każdego pracownika osobną zmienną, ale byłoby to niezwykle żmudne zajęcie. Dzięki tablicy, nie tylko szybciej możemy wprowadzić wiele imion, ale również będziemy mogli przeprowadzać na nich wiele operacji, które nie byłyby możliwe w przypadku zwykłych zmiennych. Tablicę tworzymy podobnie jak zmienne dodając przy tym charakterystyczny dla tablic nawias kwadratowy. Poniżej jeden ze sposobów stworzenia tablicy.

```
typ nazwa [ ] = {element1, element2, element3,  ... };
```

Każdy element w tablicy ma swój numer porządkowy, który nazywamy indeksem. W programowaniu indeksowanie (liczenie) zaczyna się od 0. Chcąc odwołać się do pierwszego elementu, użyjemy indeksu numer 0, chcąc odwołać się do czwartego elementu musimy posłużyć się indeksem numer 3, itd.
Weźmy pod uwagę wyraz LEGNICA, który traktujemy jako tablicę pojedynczych znaków. Każda litera jest przyporządkowana do kolejnego numeru indeksu. Pierwszy element, litera L, jest oznaczony numerem 0. Chcąc odnieść się do danego elementu, używamy właśnie numeru indeksu podając go wewnątrz nawiasu kwadratowego.

Numer indeksu	0	1	2	3	4	5	6
Element	L	E	G	N	I	C	A

Przykład

```
#include <iostream>
using namespace std;

int main() {
    string tablicaImion[] =
    {
        "Basia", "Krystian", "Adam", "Damian", "Marcin",
"Magda",
        "Darek", "Anna", "Marta", "Joanna", "Sylwia",
"Katarzyna",
        "Monika"
    };

    cout << tablicaImion [1] << endl; // Wyświetl 2-gi element
tablicy
    cout << tablicaImion [6] << endl; // Wyświetl 7-my element
tablicy
    cout << tablicaImion [9] << endl; // Wyświetl 10-ty element
tablicy
}
```

Innym sposobem utworzenia tablicy jest jej zadeklarowanie, podając w nawiasie kwadratowym ilość elementów, które będą zawarte w danej tablicy:

```
int oceny[4]; // deklaracja tablicy 4 elementowej typu int
```

Przypisanie wartości do każdego z elementów, następuje poprzez odwołanie się do niego za pomocą indeksu, np:

oceny[0] = 4;

Ten sposób może sprawdzić się w przypadku, gdy chcielibyśmy wprowadzać dane do tablicy za pomocą konsoli lub przypisać jakieś dane przy użyciu pętli.

Przykład

Utworzenie 4 elementowej tablicy typu int z ocenami

```
#include <iostream>

int main() {
    int oceny[4]; // deklaracja tablicy 4 elementowej
    oceny[0] = 4;
    oceny[1] = 5;
    oceny[2] = 2;
    oceny[3] = 6;
}
```

Wyznaczanie długości tablicy

Bardzo przydatną właściwością jest określenie długości tablicy, czyli ilości elementów w niej zawartych. W niektórych językach, istnieje metoda wykazująca w prosty sposób ilość elementów danej tablicy (często w ten sam sposób jak w przypadku zmiennych typu string). W C++ przychodzi nam radzić sobie w nieco bardziej wymagający sposób. Posługujemy się metodą, która na początku zwraca rozmiar całej tablicy - sizeof(tablica). Zwróci ona rozmiar wszystkich elementów zawartych w tablicy. Aby wyliczyć ich ilość musimy podzielić ten rozmiar, przez rozmiar pojedyńczego elementu, tj: sizeof(tablica[0]).
Nie ma znaczenia jaki numer indeksu podamy, ponieważ każdy element tablicy zajmuje dokładnie tyle samo miejsca w pamięci komputera.

Przykład

Wypisz w konsoli długość tablicy z przykładowymi ocenami

```
#include <iostream>

int main() {
    int oceny[] = { 4,6,2,4,1,3,3,5,5,6,4,2,5,3 };
    int iloscElementowWTablicy = sizeof(oceny) /
sizeof(oceny[0]);
    std::cout << iloscElementowWTablicy;
}
```

Chcąc przechować słowa lub zdania do 20 znaków, możemy to zrobić wewnątrz tablicy o typie char (jest to bardzo popularne w języku C). Tablica taka musi posiadać na końcu element dodatkowy '\0', który świadczy o zakończeniu tablicy.
W związku z powyższym tablica, która przechowuje słowo KODZIMY powinna składać się z 8 elementów.

```
char slowo[] = { 'K', 'O', 'D', 'Z', 'I', 'M', 'Y', '\0' };
```

Sortowanie tablicy

```
#include <iostream>
#include <algorithm>
using namespace std;

void wyswietlElementyTablicy(int a[]) {
     for (int i = 0; i < 5; ++i)
          cout << a[i] << " ";
}
int main() {
     int tablicaLiczb[5] = { 4, 2, 7, 9, 6 };

     cout << "\n Tablica przed sortowaniem: ";
     wyswietlElementyTablicy(tablicaLiczb);

     sort(tablicaLiczb, tablicaLiczb + 5);
     cout << "\n\n Tablica po sortowaniu rosnąco: ";
     wyswietlElementyTablicy(tablicaLiczb);

     sort(tablicaLiczb, tablicaLiczb + 5, greater < int >());
     cout << "\n\n Tablica po sortowaniu malejąco: ";
     wyswietlElementyTablicy(tablicaLiczb);
}
```

Zmiana typów

Zamiana na typ string za pomocą funkcji to_string().

```
#include <iostream>
#include <string>
using namespace std;

int main() {
     long val = 12345;
     string my_val = to_string(val);
     cout << my_val.substr(0, 2);
}
```

Zmiana tekstu na wartości liczbowe za pomocą metod:

stoi(str) – string to int

stof(str) – string to float

stol(str) – string to long

stod(str) – string to double

```
//Program wypisuje dzień urodzenia z numeru pesel

#include <iostream>
#include <string>
using namespace std;

int main(){
     string pesel = "86102625456";
     int dzien = stoi(pesel.substr(4, 2));
     cout << dzien;
}
```

Przy zmianie tekstu na liczby należy sprawdzić czy taka konwersja jest możliwa, ponieważ nie wszystkie znaki można zamienić na liczby. Możemy to przeprowadzić za pomocą mechanizmu try-catch.

```
//Zastosowanie try-catch

#include <iostream>
#include <string>
using namespace std;

int main(){
     string pesel = "86102125456";
     int dzien;

     try {
          dzien = stoi(pesel.substr(4, 2));
          cout << dzien;
     }
     catch (const exception& e) {
          cout << e.what();
     }
}
```

Wartości maksymalne danego typu

```
#include <iostream>
using namespace std;

int main() {
     cout << "Max for char: " << int(numeric_limits<int8_t>::max())
<< endl;
     cout << "Max for short: " << INT16_MAX << endl;
     cout << "Max for int: " << INT32_MAX << endl;
     cout << "Max for long: " << numeric_limits<long>::max() <<
endl;
     cout << "Max for long long: " << INT64_MAX << endl << endl;

     char a = 12;
     cout << "Rozmiar typu char(int8_t): " << sizeof(a) << endl;
     short b = 13;
     cout << "Rozmiar typu short (int16_t): " << sizeof(b) << endl;
     int c = 14;
     cout << "Rozmiar typu int (int32_t): " << sizeof(c) << endl;
     long d = 15;
     cout << "Rozmiar typu long : " << sizeof(d) << endl;
     long long e = 14;
     cout << "Rozmiar typu long long (int64_t): " << sizeof(e) <<
endl;
}
```

Enum

Typ enum służy do tego by w łatwy sposób wybierać jedną z kilku dostępnych opcji. Np. jeśli chcielibyśmy w programie wybierać pomiędzy 4 kolorami to moglibyśmy stworzyć z nich enum, tak by mieć możliwość wybrania tylko 1 z tych 4 dostępnych. Podobnie byłoby np. ze stanowiskami w pracy, ich nazwy moglibyśmy stworzyć za pomocą typu wyliczeniowego enum. Należy pamiętać jednak, że pod nazwą kryje się kolejny numer indeksu, więc enum można traktować jako słowną nakładkę dla kolejnych liczb calkowitych.

```
enum kolory {
     k_zielony, k_czerwony, k_niebieski, k_czarny
};

int main() {
     kolory mojUlubionyKolor = k_czerwony;
}
```

Struct

Struktury służą do łączenia danych w jeden obiekt. Można to zrozumieć na zasadzie przedmiotów spotykanych w życiu codziennym. Weźmy za przykład człowieka, który może mieć takie właściwości jak: imie, wzrost, waga, nr pesel, itd. Biorąc pod uwagę człowieka i jego wszystkie cechy, możemy z nich napisać instrukcję tworzenia człowieka.

```
struct czlowiek {
     string imie;
     string nrPesel;
     float wzrost;
     float waga;
};
```

Domyślnie, jeśli nie stworzymy żadnego konstruktora, to podczas tworzenia obiektu, wywoływany jest domyślny konstruktor, który nie przyjmuje żadnych argumentów.

```
     czlowiek() {

     }
```

Następnie, wewnątrz bloku struct, możemy stworzyć konstruktor, który przypisze dane domyślne, dla każdego człowieka, np.:

```
     czlowiek() {
          imie = "Anonim";
          nrPesel = "00000000000";
          wzrost = 0;
          waga = 0;
     }
```

Teraz możemy stworzyć konstruktor, który pozwoli nam wprowadzać wartości cech człowieka, tj:

```
    czlowiek(string _imie, string _nrPesel, float _wzrostCM,
float _wagaKG) {
        imie = _imie;
        nrPesel = _nrPesel;
        wzrost = _wzrostCM;
        waga = _wagaKG;
    }
```

Na koniec, na przykład, wewnątrz funkcji main możemy stworzyć człowieka domyślnego lub takiego z konkretnymi cechami:

```
int main() {
    system("chcp 1250>nul");
    czlowiek ludek1;
    czlowiek ludek2("Janek", "88102309091", 178, 72);
}
```

Po stworzeniu człowieka możemy zmieniać jego cechy lub je wyświetlać, np.:

```
    ludek2.waga = 76;
    cout << ludek2.imie << " waży teraz " << ludek2.waga <<
"kg.";
```

Przykładowe użycie struktury w języku C++, pełny program:

```
#include <iostream>
using namespace std;

struct czlowiek {
	string imie;
	string nrPesel;
	float wzrost;
	float waga;

	czlowiek() {
		imie = "Anonim";
		nrPesel = "00000000000";
		wzrost = 0;
		waga = 0;
	}

	czlowiek(string _imie, string _nrPesel, float _wzrostCM,
float _wagaKG) {
		imie = _imie;
		nrPesel = _nrPesel;
		wzrost = _wzrostCM;
		waga = _wagaKG;
	}
};

int main() {
	system("chcp 1250>nul");
	czlowiek ludek1;
	czlowiek ludek2("Janek", "88102309091", 178, 72);
	ludek2.waga = 76;
	cout << ludek2.imie << " waży teraz " << ludek2.waga <<
"kg.";
}
```

Klasa

Klasa to rozszerzona struktura danych, która jest istotą programowania obiektowego. Służy do tego by za pomocą prostych wartości określić obiekty złożone. Tworząc nową klasę, tworzy się nowy typ zmiennej, która służy w określonym celu. Na początek rozważmy sytuację, w której potrzebujemy programu, który przetwarza informacje związane z prostokątami. Stworzymy klasę o nazwie prostokąt (poza funkcją main) i podamy cechy takie jak szerokosc i wysokosc.

```
class prostokat {
     float szerokosc, wysokosc;
}
```

Następnie stworzymy konstruktor publiczny, który pozwoli przypisać dane do tworzonego prostokąta:

```
class prostokat {
     float szerokosc, wysokosc;
public:
     prostokat(float _szer, float _wys) {
          szerokosc = _szer;
          wysokosc = _wys;
     }
};
```

Teraz w funkcji main możemy stworzyć prostokąt o konkretnej szerokości i wysokości.

```
int main() {
     prostokat p1(10, 6);
}
```

Te dane jednak niewiele nam dają, załóżmy, że po stworzeniu prostokąta chcemy wyświetlić jego pole. Stwórzmy zatem odpowiednią funkcję wewnątrz klasy prostokat.

```
#include <iostream>
using namespace std;

class prostokat {
     float szerokosc, wysokosc;
public:
     prostokat(float _szer, float _wys) {
          szerokosc = _szer;
          wysokosc = _wys;
     }

     void wyswietlPole() {
          cout << "Pole wynosi: " << szerokosc * wysokosc;
     }
};

int main() {
     prostokat p1(10, 6);
     p1.wyswietlPole();
}
```

Możemy też zwrócić wartość pola prostokąta czy obwodu za pomoca osobnych funkcji, np.:

```
     float obliczPole() {
          return szerokosc * wysokosc;
     }

     float obliczObwod() {
          return 2 * szerokosc + 2 * wysokosc;
     }
```

Możemy też narysować taki prostokąt, a ostateczny mógłby wyglądać w poniższy sposób:

```
#include <iostream>
using namespace std;
class prostokat {
     float szerokosc, wysokosc;
public:
     prostokat(float _szer, float _wys) {
          szerokosc = _szer;
          wysokosc = _wys;
     }

     float obliczPole() {
          return szerokosc * wysokosc;
     }

     float obliczObwod() {
          return 2 * szerokosc + 2 * wysokosc;
     }

     void wyswietlPoleiObwod() {
          cout << "Pole wynosi: " << obliczPole() << endl;
          cout << "Obwód wynosi: " << obliczObwod() << endl;
     }

     void rysuj() {

          for (int i = 0; i < int(wysokosc); i++) {
               for (int j = 0; j < int(szerokosc); j++) {
                    cout << " * ";
               }
               cout << endl;
          }
     }
};

int main() {
     system("chcp 1250>nul");
     prostokat p1(10, 6);
     p1.wyswietlPoleiObwod();
     p1.rysuj();
}
```

Wskaźniki

Każda zmienna czy obiekt tworzony w czasie działania programu ma pewną wartość i zajmuje określoną ilość pamięci podręcznej. Wartość takiej zmiennej jest zapisywana w określonym miejscu, którego zazwyczaj nie musimy znać, ponieważ wartości te wywołujemy za pomocą nazwy zmiennej. Czasami jednak warto posługiwać się adresami takich zmiennych, zamiast samymi wartościami. W takich przypadkach możemy stworzyć wskaźnik do zmiennej, czyli zmienną, która zamiast prowadzić nas bezpośrednio do wartości, przechowuje adres.

Wskaźnik tworzy się podobnie jak zwykłą zmienną, ale dodaje się znak *, a podczas przypisywania zmiennej poprzedza się ją znakiem &.

```
int nrDomu = 34;
int* p_nrDomu = &nrDomu;
```

Od teraz wskaźnik p_nrDomu będzie odnosił się do adresu zmiennej nrDomu. Aby wyświetlić adres zmiennej nrDomu możemy użyć wskażnika, co zwróci wartość w kodzie szesnastkowym:

```
int nrDomu = 34;
int* p_nrDomu = &nrDomu;
cout << p_nrDomu;
```

Aby wyświetlić wartość do której adresu odnosi się wskaźnik, należy przed jego nazwą użyć *(który czasem jest nazywany operatorem wyłuskania)

```
cout << *p_nrDomu;
```

Innym sposobem wyświetlenia, czy operowania na adresach, bez użycia wskaźników, jest użycie referencji, czyli znaku &.

```
int nrDomu = 34;
cout << &nrDomu;
```

Przykład użycia wskaźników w programie do zamiany wartości zmiennych

```
#include <iostream>
using namespace std;

void zamien(int* x, int* y){
    int  pom = *x;
    *x = *y;
    *y = pom;
}

int main(){
    system("chcp 1250>nul");
    int x, y;
    cout << "Podaj x:";
    cin >> x;
    cout << "Podaj y:";
    cin >> y;
    zamien(&x, &y); //przekazujemy adresy zmiennych
    cout << "Wartości zmiennych zostały zamienione, x = "<< x <<
", y = " << y;
}
```

```
#include <iostream>
using namespace std;

int main() {
     system("chcp 1250>nul");
     int x;          //Zwykła zmienna całkowita
     int* w_int; //Wskaźnik do zmiennej całkowitej
     w_int = &x; //Przypisz adres zmiennej x do wskaźnika *w_int
     cout << " Podaj liczbę : ";
     cin >> x; //Wczytaj wartość do x; można tu także  zapisu *w_int
     cout <<"Odczytanie wartości z pamięci wskazywanej przez
wskaźnik: " <<  *w_int << "\n";
     cout << "Adres zmiennej x: " << w_int << endl;
     * w_int = 10;  // Zmiana wartości w miejscu wskazywanym przez
wskaźnik
     cout << "Wartość x po zmianie za pomocą wskaźnika: " << x; //
Teraz wyświetli 10
}
```

Sekcja z zadaniami

Pole trójkąta

Napisz program, który prosi o podanie podstawy i wysokości trójkąta, a następnie wyświetla jego pole.

```
#include <iostream>
using namespace std;
int main() {
    system("chcp 1250>nul");
    float podstawa, wysokosc, pole;
    cout << "Program do obliczania pola powierzchni trójkąta."
<< endl;
    cout << "Podaj długość podstawy trójkąta: ";
    cin >> podstawa;
    cout << "Podaj wysokość trójkąta: ";
    cin >> wysokosc;
    pole = podstawa * wysokosc / 2;
    cout << "Pole trójkąta wynosi: " << pole;
}
```

Przeliczanie calów na centymetry

Napisz program, który przelicza cale na centymetry.

1 cal to 2.54 cm.

```
#include <iostream>
using namespace std;
int main() {
    system("chcp 1250>nul");
    float wartoscWCalach;
    cout << "Program do przeliczania cali na centymetry" <<
endl;
    cout << "Podaj wartość w calach: ";
    cin >> wartoscWCalach;
    cout << wartoscWCalach << " cali, to " << wartoscWCalach *
2.54 << " centymetrów.";
}
```

Losowa liczba z przedziału

Napisz program, który losuje losową liczbę z przedziału 2-9. Jeśli wylosowna liczba będzie mniejsza od 5, wyświetl komunikat "Liczba mniejsza od 5". W przeciwnym razie wyświetl komunikat "Liczba większa lub równa 5".

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
    system("chcp 1250>nul");
    srand(time(0));
    int losowaLiczba = (rand() % 8) + 2;
    cout << losowaLiczba << endl;
    if (losowaLiczba < 5) {
        cout << "Liczba mniejsza od 5";
    }
    else {
        cout << "Liczba większa lub równa 5";
    }
}
```

Dodawanie liczb

Napisz program, który dodaje dwie, podane jako argumenty, liczby i wyświetla ich wynik w konsoli. Stwórz dodatkową funkcję o nazwie dodajLiczby().

```
#include <iostream>
using namespace std;

void dodajLiczby(int l1, int l2) {
    cout << l1 << " + " << l2 << " = " << l1 + l2;
}

int main() {
    dodajLiczby(40, 20);
}
```

Dodawanie liczb pobranych z konsoli

Napisz program, który pobiera dwie liczby z konsoli, a następnie wyświetla ich sumę.

Zrób to za pomocą osobnych funkcji, np: pobierzLiczbe() i dodajLiczby()

```
#include <iostream>
using namespace std;

int pobierzLiczbe(int nrPorzadkowy) {
    cout << "Podaj " << nrPorzadkowy << " liczbę:";
    int liczba;
    cin >> liczba;
    return liczba;
}

void dodajLiczby(int liczba1, int liczba2) {
    cout << liczba1 << " + " << liczba2 << " = " << (liczba1 +
liczba2);
}

int main() {
    system("chcp 1250>nul");
    int licz1 = pobierzLiczbe(1);
    int licz2 = pobierzLiczbe(2);
    dodajLiczby(licz1, licz2);
    return 0;
}
```

Generowanie losowej litery

Napisz program, który będzie generował losową literę od A do Z.

A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Na początek sprawdzamy w tabeli ASCII zakres liczb w którym dostępne są litery od A do Z. Wiemy już, że litery te są w zakresie od 65 do 90.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int generujLosowaLiczbe() {
 return rand() % (90 - 65 + 1) + 65;
}

void wyswietlLitere(int losowaLiczba) {
 cout << "Losowa litera: " << char(losowaLiczba);
}

int main() {
 srand(time(0));
 wyswietlLitere(generujLosowaLiczbe());
}
```

Generowanie losowego pola(np. a2, b6, itp.)

Napisz program, który wyświetla losową liczbę w zakresie <1,10> oraz małą literę <a,j>, a następnie wyświetla ich połączenie, np: a2, b6, d7. Mógłby to być pierwszy krok do stworzenia komputerowej gry w statki :)

a	97
b	98
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int generujLosowaLiczbe() {
    return rand() % 10 + 1;
}

char generujLosowaLitera() {
    return rand() % (106 - 97 + 1) + 97;
}

int main() {
    srand(time(0));
    cout << "Losowe pole: " <<
generujLosowaLitera() << generujLosowaLiczbe();
}
```

Wyświetlanie znaków z tekstu

Pętla for może być użyta w przypadku tablic jak i wartości tekstowych w prostszej formie.
Napisz program, który wyświetli, w kolumnie, każdy znak z wartości typu string.

2 warianty:

I

```
#include <iostream>
using namespace std;

int main() {
     string a = "C++ jest ok";
     for (int i = 0; i < a.length(); i++) {
          cout << a[i] << endl;
     }
}
```

II

```
#include <iostream>
using namespace std;
int main() {
    string a = "C++ jest ok";

    for (char i : a) {
        cout << i << endl;
    }
}
```

Wypisz liczby: 0, 10, 20, ..., 100

Program, który wypisze kolejne 10-tki, w przedziale od 0 do 100 tj.(0,10,20,30...)

```
#include <iostream>
using namespace std;

int main() {

	// za pomocą pętli for
	for (int i = 0; i <= 100; i += 10) {
		cout << i << " ";
	}


	// za pomocą pętli while
	int i = 0;
	while (i <= 100) {
		cout << i << " ";
		i += 10;
	}
}
```

Wypisz liczby: 10,9,8,...,0

Program, który wyświetli liczby od 10 do 0, w jednej linii, oddzielonych przecinkami.
Następnie, zmodyfikuj program, aby przecinek nie pokazał się po cyfrze 0.

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 10; i >= 0; i--) {
        cout << i << ", ";
    }
}
```

Zmodyfikuj program, aby przecinek nie pokazał się po cyfrze 0.

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 10; i >= 0; i--) {
        if (i > 0) {
            cout << i << ", ";
        }
        else {
            cout << i;
        }
    }
}
```

Wyświetl tabliczkę mnożenia

Napisz program, który wyświetli pierwszą kolumnę tabliczki mnożenia.

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i <= 10; i++) {
        cout << "1 x " << i << " = " << i << endl;
    }
}
```

Napisz program, który wyświetli całą tabliczkę mnożenia.

```
#include <iostream>
using namespace std;

int main() {
     for (int i = 1; i <= 10; i++) {
          for (int j = 1; j <= 10; j++) {
               cout << i << " x " << j << " = " << i * j<< endl;
          }
     }
}
```

Wyświetl losową osobę z tablicy

Napisz program, który losowo wybiera osobę z tablicy i wyświetla jej imię.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main() {
    srand(time(0));
    string osoby[] = {
        "Basia", "Krystian", "Adam", "Damian", "Marcin",
"Magda", "Darek",
        "Anna", "Marta", "Joanna", "Sylwia", "Katarzyna",
"Monika"
    };

    int rozmiarTablicy = sizeof(osoby) / sizeof(osoby[0]);
    int losowaLiczba = rand() % rozmiarTablicy;
    cout << osoby[losowaLiczba];
}
```

Losowe działanie z wprowadzaniem odpowiedzi

Napisz program, który wyświetli losowe zadanie mnożenia. W zależności od poprawności udzielonej odpowiedzi wyświetli odpowiedni komunikat. W tym przykładzie użyj zmiennych globalnych oraz 3 funkcji dodatkowych do wyświetlenia działania, pobrania odpowiedzi i jej sprawdzenia.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int liczba1, liczba2, odp;

void podajOdpowiedz() {
     cin >> odp;
}

void sprawdzOdpowiedz() {
     if (odp == liczba1 * liczba2) {
          cout << "Brawo!";
     }
     else {
          cout << " Odpowiedź niepoprawna";
     }
}

void wyswietlDzialanie() {
     liczba1 = rand() % 10 + 1;
     liczba2 = rand() % 10 + 1;
     cout << liczba1 << " * " << liczba2 << " = ";
}

int main() {
     system("chcp 1250>nul");
     srand(time(0));
     wyswietlDzialanie();
     podajOdpowiedz();
     sprawdzOdpowiedz();
}
```

Podaj promień koła, wyświetl obwód i pole koła (zmienne lokalne)

Napisz program, który pobiera od użytkownika promień koła, a następnie wyświetla jego obwód i pole. Użyj osobnych funkcji do pobrania promienia, obliczenia obwodu i pola. Nie korzystaj ze zmiennych globalnych.

```
#include <iostream>
using namespace std;

float pobierzPromien() {
    cout << "Podaj promień koła: ";
    float promien;
    cin >> promien;
    return promien;
}

float obliczObwod(float promien) {
    float obwod = 2 * 3.14 * promien;
    return obwod;
}

float obliczPole(float promien) {
    float pole = 3.14 * promien * promien;
    return pole;
}

int main() {
    system("chcp 1250>nul");
    float promien = pobierzPromien();
    cout << "Obwód wynosi: " << obliczObwod(promien) << endl;
    cout << "Pole wynosi: " << obliczPole(promien) << endl;
}
```

Wyświetl liczby od 14 do 33, poza liczbą 20 i 30

Wyświetl liczby od 14 do 33, poza liczbą 20 i 30. Możesz skorzystać z opcji continue, spróbuj znaleźć informacje na jej temat w internecie (wpisz for loop continue c++).

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 14; i <= 33; i++) {
        if (i != 20 && i != 30) {
            cout << i << " ";
        }
    }
}
```

lub z opcją continue:

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 14; i <= 33; i++) {
        if (i == 20 || i == 30) {
            continue;
        }
        else {
            cout << i << " ";
        }
    }
}
```

Przypisz do tablicy 10 losowych liczb i wyświetl rosnąco

Napisz program, który przypisuje do tablicy 10 losowych liczb z przedziału od 0 - 100, sortuje je i wyświetla w jednej linii.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <algorithm>
using namespace std;

int main() {
	srand(time(0));
	int liczby[10];

	int dlugoscTablicy = sizeof(liczby) / sizeof(liczby[0]);
	for (int i = 0; i < dlugoscTablicy; i++) {
		liczby[i] = rand() % 100;
	}

	sort(liczby, liczby + dlugoscTablicy);

	for (int i = 0; i < dlugoscTablicy; i++) {
		cout << liczby[i] << " ";
	}
}
```

Pobierz 3 liczby i wyświetl największą za pomocą tablicy

Napisz program, który pobiera 3 liczby, po czym wyświetla największą z nich.

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    system("chcp 1250>nul");
    int liczby[3];
    for (int i = 1; i < 4; i++) {
        cout << "Podaj liczbę " << i << ": ";
        cin >> liczby[i-1];
    }
    int dlugoscTablicy = sizeof(liczby) / sizeof(liczby[0]);
    sort(liczby, liczby + dlugoscTablicy, greater < int >());
    cout << "Największa z podanych liczb to: " << liczby[0];
}
```

Podaj wymiary akwarium, wyświetl kubaturę

Napisz program, który pobiera długości boków akwarium w centymetrach, a następnie wyświetla jego kubaturę w litrach.

```
#include <iostream>
using namespace std;

int main() {
    system("chcp 1250>nul");
    float szerokosc;
    float glebokosc;
    float wysokosc;
    float kubatura;

    cout << "Podaj szerokość w cm: ";
    cin >> szerokosc;
    cout << "Podaj głębokość w cm: ";
    cin >> glebokosc;
    cout << "Podaj wysokość w cm: ";
    cin >> wysokosc;
    kubatura = szerokosc * glebokosc * wysokosc;
    cout << "Kubatura podanego akwarium to: " << kubatura /
1000 << " litrów.";
}
```

Po refaktoryzacji kodu, z dodaniem funkcji:

```
#include <iostream>
using namespace std;

float wprowadzDane(string info) {
    cout << info;
    int odp;
    cin >> odp;
    return odp;
}

int main() {
    system("chcp 1250>nul");
    float kubatura;
    kubatura = wprowadzDane("Podaj szerokość w cm: ") *
wprowadzDane("Podaj głębokość w cm: ") * wprowadzDane("Podaj
wysokość w cm: ");
    cout << "Kubatura podanego akwarium to: " << kubatura /
1000 << " litrów.";
}
```

Zabawne połączenie zdań z dwóch tablic

Napisz program, który będzie wyświetlał połączenia zdań z dwóch tablic. Umieść w jednej tablicy rzeczowniki, a w drugiej dowolne zdania, co w połączeniu będzie wyświetlać zabawne sentencje. Elementy z obu tablic powinny być wybierane losowo, np: Alex ma 4 krawędzie.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main() {
    system("chcp 1250>nul");
    srand(time(0));

    string tab1[] = {
        "Laptop", "Andrzej", "Rower", "Kotek", "Śrubokręt",
"Parasolka", "Ziemniak", "Rosół" };

    string tab2[] = {
        "ma 21 cali.", "ma długie wąsy.", "zimuje w piwnicy.",
"smakuje najlepiej z pietruszką.", "chroni przed deszczem." };

    int dlugoscTab1 = sizeof(tab1) / sizeof(tab1[0]);
    int dlugoscTab2 = sizeof(tab2) / sizeof(tab2[0]);

    int rand1 = rand() % dlugoscTab1;
    int rand2 = rand() % dlugoscTab2;

    cout << tab1[rand1] << " " << tab2[rand2];
}
```

Zabawne połączenie zdań z dwóch tablic

Napisz program, który będzie wyświetlał połączenia zdań z dwóch tablic. Umieść w jednej tablicy rzeczowniki, a w drugiej dowolne zdania, co w połączeniu będzie wyświetlać zabawne sentencje. Elementy z obu tablic powinny być wybierane losowo, np: Alex ma 4 krawędzie. Program ma działać ciągle, dopóki nie zostanie wciśnięty klawisz z. Kolejne losowe zdanie ma się wyświetlić po wciśnięciu klawisza n.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main() {
    system("chcp 1250>nul");
    srand(time(0));

    string tab1[] = {
        "Laptop", "Andrzej", "Rower", "Kotek", "Śrubokręt",
"Parasolka", "Ziemniak", "Rosół" };

    string tab2[] = {
        "ma 21 cali.", "ma długie wąsy.", "zimuje w piwnicy.",
"smakuje najlepiej z pietruszką.", "chroni przed deszczem." };

    int dlugoscTab1 = sizeof(tab1) / sizeof(tab1[0]);
    int dlugoscTab2 = sizeof(tab2) / sizeof(tab2[0]);

    char znak = 'n';
    cout << "Podaj n (następny) lub z (zakończ)" << endl;
    do {
        if (znak == 'n') {
            int rand1 = rand() % dlugoscTab1;
            int rand2 = rand() % dlugoscTab2;
            cout << tab1[rand1] << " " << tab2[rand2];
            cin >> znak;
        }
        else {
            cout << "Podaj n (następny) lub z (zakończ)";
            cin >> znak;
        }
    } while (znak != 'z');
}
```

5 działań matematycznych z oceną (zmienne lokalne)

Napisz program, który będzie wyświetlał 5 losowych działań matematycznych, sprawdzał poprawność naszych odpowiedzi i przydzielał punkty. Po udzieleniu ostatniej odpowiedzi, program się zakończy i wyświetli procent poprawnych odpowiedzi. W tym przykładzie nie używaj zmiennych globalnych.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int generujLosowaLiczbe() {
    return rand() % 20 + 1;
}

void wyswietlDzialanie(int l1, int l2) {
    cout << l1 << " + " << l2 << " = ";
}

int pobierzOdpowiedz() {
    int odpUzytkownika;
    cin >> odpUzytkownika;
    return odpUzytkownika;
}

int sprawdzOdpowiedz(int l1, int l2, int odp) {
    if (odp == l1 + l2) {
        return 1;
    }
    else {
        return 0;
    }
}

int main() {
    system("chcp 1250>nul");
    int liczbaPunktow = 0;
    srand(time(0));
    int liczba1, liczba2, odp;

    for (int i = 0; i < 5; i++) {
        liczba1 = generujLosowaLiczbe();
        liczba2 = generujLosowaLiczbe();
        wyswietlDzialanie(liczba1, liczba2);
        odp = pobierzOdpowiedz();
        liczbaPunktow += sprawdzOdpowiedz(liczba1, liczba2,
odp);
    }
    float wynik = (float)liczbaPunktow / 5.0;
    cout << "Twój wynik to: " << wynik * 100 << " %";
}
```

Książka

Napisz program, książkę. Po wyświetleniu danej strony, wyświetli zapytanie o gotowość przejścia do kolejnej lub poprzedniej strony. Po wpisaniu 2, przejdzie do kolejnej strony, a po wciśnięciu 1 wróci do poprzedniej. Po wciśnięciu 0, program się zakończy.

```
#include <iostream>
using namespace std;
string ksiazka[6];
int strona;
int userInput;

void wyswietlStrone() {
    if (userInput == 2 && strona < 5) {
        strona++;
    }
    else if (userInput == 1 && strona > 0) {
        strona--;
    }
    else {
        strona = 0;
    }

    cout << "Wpisz 2 by przejść dalej, 1 by wrócić lub 0 by
zakończyć program" << endl;
    cout << "Strona " << strona << endl << ksiazka[strona] << endl
<< endl;
    cin >> userInput;
}
int main() {
    system("chcp 1250>nul");
    strona = 0;
    userInput = 3;
    ksiazka[0] = "\t Opowieść o 4 bitach.";
    ksiazka[1] = "\t Pierwszy bit miał na imię Array. Uwielbiał
gromadzić przedmioty tego samego typu. Z czasem nauczył się je
układać i sortować na wiele sposobów.";
    ksiazka[2] = "\t Drugi bit miał na imię Loop. Posiadł umiejętność
wykonywania powtarzalnych czynności.";
    ksiazka[3] = "\t Trzeci bit miał na imię Switch. Miał siostrę z
którą na zmianę podejmowali ważne decyzje w swoim królestwie.";
    ksiazka[4] = "\t Czwarty bit nazywał się Zmienną. Miał wiele
typów osobowości i często zmieniał się w zależności od sytuacji.";
    ksiazka[5] = "\t Koniec";

    while (userInput != 0) {
        wyswietlStrone();
    }
    cout << "Dziękujemy za przeczytanie książki.\n Do widzenia!";
}
```

Elementy języka C

Wyprowadzanie danych w C – printf()

Wyprowadzanie danych następuję za pomocą metody printf(). Jako argumenty należy podać specyfikator formatowania wewnątrz cudzysłowu, a następnie jako drugi argument podaje się zmienną lub wartość do wyświetlenia. Argumenty oddziela się przecinkiem. Wymagane nagłówki:

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

Specyfikatory formatowania

%d	– liczba całkowita	`printf("%d", 123);`
%f	– liczba dziesiętna float lub double	`printf("%f", 12.43);`
%.nf	– n cyfr dziesiętnych	`printf("%.2f", 12.4334);`
%N.nf	– N (szerokość pola), n (ilość cyfr po przecinku)	`printf("%20.3f", 12.4345);`
%-N.nf	– wyrównywanie liczby do lewej strony	`printf("%-20.3f", 12.4345);`
%c	– znak	`printf("%c", '@');`
%s	– łańcuch	`printf("%s", "Hello");`
%e	– liczba w postaci wykładniczej	`printf("%e", 120040.4345);`
%.ne	– liczba w postaci wykładniczej z n dokładnością	`printf("%.3e", 120040.4345);`
%ns	– łańcuch po prawej stronie pola n-znakowego	`printf("%20s", "Hello");`
%-ns	– łańcuch po lewej stronie pola n-znakowego	`printf("%-20s", "Hello");`
%o	– liczba ósemkowa	
%x	– liczba szesnastkowa	
%u	– liczba bez znaku	
%h	– liczba krótka (short int lub unsigned short int)	
%g	– liczby rzeczywiste w najkrótszym zapisie dziesiętnym lub wykładniczym	
%0N.nf	– wypełnienie miejsca z lewej strony zerami	
%+N.nf	– liczba poprzedzona znakiem + lub –	

Przykład

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

int main() {
     system("chcp 1250>nul");
     char znak;
     znak = '@';
     printf("Znak: %c\n", znak);
     printf("ASCII DEC: %d\n", znak);
     printf("ASCII HEX: %x\n", znak);
     getchar();
}
```

Wczytywanie danych w C - getchar(), scanf()

Wymagane headery:

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

getchar()

Metoda getchar() pobiera pojedynczy znak z klawiatury, który zazwyczaj będziemy przypisywać do zmiennej typu char. Należy pamiętać, że nawet jeśli, ktoś wpisze więcej znaków, to pobrany zostanie tylko jeden, pierwszy wprowadzony znak.

Przykład

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

int main() {
     system("chcp 1250>nul");
     printf("Wpisz znak: ");
     char znak = getchar();
     printf("\nPodales: %c", znak);
}
```

scanf()

Używając funkcji scanf() należy podać 2 argumenty, pierwszy to zapis formatowania (patrz sekcja z printf), mówiący o tym jaką wartość pobieramy, a drugi to nazwa zmiennej do której ta wartość ma zostać przypisana. W przypadku wartości numerycznych i char, należy podać adres zmiennej, czyli przed nazwą zmiennej dopisuje się znak &.

Przykład 1

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

int main() {
     system("chcp 1250>nul");
     printf("Podaj liczbę: ");
     int liczba;
     scanf("%d", &liczba);
     printf("\nPodales: %d",
liczba);
     return 0;
}
```

Przykład 2

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

int main() {
     system("chcp 1250>nul");
     printf("Podaj swoje imię:
");
     char imie[20];
     scanf("%s", imie);
     printf("\nCześć %s",
imie);
     return 0;
}
```

Zadania C / C++

Obliczanie drogi i prędkości w ruchu jednostajnie przyśpieszonym

Program, który wczytuje parametry ruchu jednostajnie przyspieszonego (a i vo) i oblicza przebytą drogę oraz prędkość po czasie t.

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

int main() {
    int a, v, t;
    float s;
    printf("Podaj przyspieszenie a: ");
    scanf("%d", &a);
    printf("Podaj predkosc poczatkowa v: ");
    scanf("%d", &v);
    printf("Podaj czas t: ");
    scanf("%d", &t);
    s = (v * t) + (a * t * t) / 2.0;

    printf("Droga s wynosi :%5.2f\n", s);
    system("PAUSE");
    return 0;
}
```

Kody ASCII dziesiętnie i szesnastkowo

Program, który wczytuje znak z klawiatury, wyświetla znak oraz jego kod ASCII dziesiętnie i szesnastkowo.

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

int main() {
     char znak;
     printf("Podaj znak: ");
     znak = getchar();
     printf("Znak: %c\n", znak);
     printf("ASCII DEC: %d\n", znak);
     printf("ASCII HEX: %x\n", znak);
}
```

Liczba dziesiętna w postaci ósemkowej i szesnastkowej

Program, który wczytuje liczbę całkowitą dziesiętną i wyświetla ją w postaci dziesiętnej, ósemkowej oraz szesnastkowej.

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

int main() {
     system("chcp 1250>nul");
     int liczba;
     printf("Podaj liczbę całkowitą: ");
     scanf("%d", &liczba);
     printf("DEC: %d\n", liczba);
     printf("OCT: %o\n", liczba);
     printf("HEX: %x\n", liczba);
}
```

Obliczanie pola trójkąta prostokątnego

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {
     system("chcp 1250>nul");
     float a, b;
     printf("Podaj pierwszą przyprostokątną: ");
     scanf("%f", &a);
     printf("\nPodaj drugą przyprostokątną: ");
     scanf("%f", &b);
     float pole = a * b / 2;
     printf("\nPole trójkąta wynosi: %.2f", pole);
     float obwod = sqrt(pow(a, 2) + pow(b, 2)) + a + b;
     printf("\nObwód trójkąta wynosi: %.2f", obwod);
}
```

Rezystancja dwóch rezystorów połączonych równolegle

Program obliczający rezystancję zastępczą dwóch rezystorów połączonych równolegle.

Wynik wyświetlić w postaci wykładniczej (trzy cyfry znaczące).

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main() {
     float r1, r2, rz;
     printf("\nPodaj rezystancję R1: ");
     scanf("%f", &r1);
     printf("\nPodaj rezystancję R2: ");
     scanf("%f", &r2);
     rz = pow(pow(r1, -1) + pow(r2, -1), -1);
     printf("Rezystancja zastępcza: %.2f Ohm\n", rz);
     printf("Rezystancja zastępcza w not.wykł: %.2e", rz);
}
```

Wyznacz wartość maksymalną z 3 podanych liczb

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
     system("chcp 1250>nul");
     int a, b, c, max;
     printf("Podaj pierwszą liczbę,a : ");
     scanf("%d", &a);
     printf("Podaj drugą liczbę,b:  ");
     scanf("%d", &b);
     printf("Podaj trzecią liczbę,c: ");
     scanf("%d", &c);

     max = a;

     if (b > a) {
          max = b;
     }

     if (c > b && c > a) {
          max = c;
     }
     printf("Maks wynosi: %d", max);
}
```

Wyświetl dzień tygodnia na podstawie liczby

W oparciu o wprowadzoną liczbę Dzien [1..7] wyświetlić słownie nazwę dnia tygodnia używając instrukcji warunkowej

```
#include <stdio.h>
#include <stdlib.h>
int main() {
     system("chcp 1250>nul");

     int liczba;
     printf("Podaj nr od 1 do 7: ");
     scanf("%d", &liczba);
     if (liczba == 1) {
          printf("Poniedziałek");
     }
     else if (liczba == 2) {
          printf("Wtorek");
     }
     else if (liczba == 3) {
          printf("Środa");
     }
     else if (liczba == 4) {
          printf("Czwartek");
     }
     else if (liczba == 5) {
          printf("Piątek");
     }
     else if (liczba == 6) {
          printf("Sobota");
     }
     else if (liczba == 7) {
          printf("Niedziela");
     }
     else {
          printf("Wprowadzono złą wartość");
     }
}
```

Sprawdzić, czy liczba całkowita X jest jednocześnie podzielna przez 7 oraz 3.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
     system("chcp 1250>nul");
     int liczba;
     puts("Podaj liczbę całkowitą: ");
     scanf("%d", &liczba);
     if (liczba % 7 == 0 && liczba % 3 == 0) {
          printf("Liczba %d jest podzielna przez 7 i 3",
liczba);
     }
     else {
          printf("Liczba %d nie jest podzielna przez 7 i 3",
liczba);
     }
}
```

Obliczanie pierwiastków równania kwadratowego

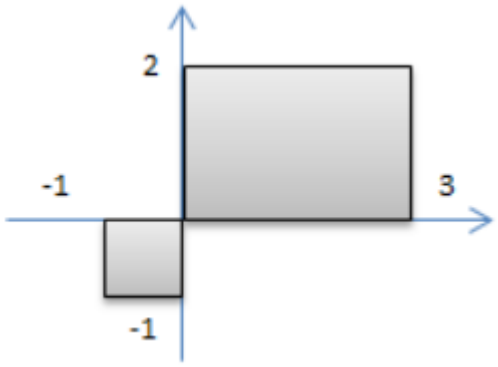
Dano równanie kwadratowe:ax2 + bx + c = 0 . Wprowadzić wartości float a,b,c z klawiatury(sprawdzić wartość a czy jest 0), a następnie obliczyć DELTĘ i w zależności od jej wartości wyświetlić komunikat (BRAK PIERWIASTKÓW),jest jeden pierwiastek i obliczyć jego wartość x1 oraz wyświetlić, są dwa pierwiastki równania x1 oraz x2 i obliczyć oraz wyświetlić wartości.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
int main() {
	system("chcp 1250>nul");
	float a, b, c, d, x, x1, x2, pd;
	printf("Wprowadź zmienną a: ");
	scanf("%f", &a);
	if (a == 0) {
		printf("To nie jest równanie kwadratowe");
		return 0;
	}
	printf("Wprowadź zmienną b: ");
	scanf("%f", &b);
	printf("Wprowadź zmienną c: ");
	scanf("%f", &c);
	d = b * b - 4 * a * c;
	printf("Delta wynosi %.2f\n", d);

	if (d < 0) {
		printf("Delta < 0, Równanie nie ma rozwiązań");
		return 0;
	}
	else {
		pd = sqrt(d);

		if (d == 0.0) {
			x = (-b) / 2 * a;
			printf("Delta = 0, Równanie ma 1 rozwiązanie, x = %.2f",
x);

		}
		else if (d > 0) {
			x1 = (-b - pd) / 2 * a;
			x2 = (-b + pd) / 2 * a;
			printf("Delta > 0, Równanie ma 2 rozwiązanie\n");
			printf("x1 = %.2f, x2 = %.2f", x1, x2);
		}
	}
}
```

Punkt na płaszczyźnie

Dano punkt na płaszczyźnie P(x,y).

Napisać program, który sprawdzi, czy dany punkt należy do wskazanego obszaru.




```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

int main() {
     system("chcp 1250>nul");
     float x, y;
     printf("podaj współrzędne x punktu P: ");
     scanf("%f", &x);
     printf("podaj współrzędne y punktu P: ");
     scanf("%f", &y);
     if ((x >= -1 && y >= -1 && x <= 0 && y <= 0) || (x >= 0 &&
y >= 0 && x <= 3 && y <= 2)) {
          printf("Punkt P= %.2f,%.2f należy do wskazanego
obszaru", x, y);
     }
     else {
          printf("Punkt P= %.2f,%.2f nie należy do wskazanego
obszaru", x, y);
     }
}
```

Liczba dni wybranego miesiąca

Napisz program, który na podstawie podanego numeru miesiąca wyświetli ile ma dni.

// Styczeń, Marzec, Maj, Lipiec, Sierpień, Październik, Grudzien - 31 dni

// Luty - 28 dni

// Kwiecień, Czerwiec, Wrzesień, Listopad - 30 dni

Wersja I

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

int main() {
     system("chcp 1250>nul");
     int miesiac;
     printf("Podaj nr miesiąca: ");
     scanf("%d", &miesiac);
     switch (miesiac) {
     case 1: printf("31 dni"); break;
     case 2: printf("28 dni"); break;
     case 3: printf("31 dni"); break;
     case 4: printf("30 dni"); break;
     case 5: printf("31 dni"); break;
     case 6: printf("30 dni"); break;
     case 7: printf("31 dni"); break;
     case 8: printf("31 dni"); break;
     case 9: printf("30 dni"); break;
     case 10: printf("31 dni"); break;
     case 11: printf("30 dni"); break;
     case 12: printf("31 dni"); break;
     default: printf("Nieprawidłowy numer"); break;
     }
}
```

Wersja II

```
#pragma warning(disable : 4996)
#include <stdlib.h>
#include <stdio.h>

int main() {
     system("chcp 1250>nul");
     int miesiac;
     printf("Podaj nr miesiąca: ");
     scanf("%d", &miesiac);
     switch (miesiac) {
     case 1: case 3: case 5: case 7: case 8: case 10: case 12:
          printf("31 dni"); break;
     case 2:
          printf("28 dni"); break;
     case 4: case 6: case 9: case 11:
          printf("30 dni"); break;
     default:
          printf("Nieprawidłowy numer"); break;
     }
}
```

Pora roku na podstawie dnia i miesiąca

Wprowadź numer dnia i miesiąca; na ich podstawie wyświetl porę roku.

```
#pragma warning(disable : 4996) //wyłączenie błędów dla Visual Studio
#include <stdlib.h>
#include <stdio.h>

//Wiosna: 21 marzec do 20 czerwiec
//Lato : 21 czerwiec do 20 wrzesień
//Jesień : 21 wrzesień do 20 grudzień
//Zima : 21 grudzień do 20 marzec

int main() {
	system("chcp 1250>nul");
	int dzien, miesiac;
	printf("Podaj dzień: ");
	scanf("%d", &dzien);
	printf("Podaj miesiąc: ");
	scanf("%d", &miesiac);

	if (dzien >= 21 && miesiac == 3 || miesiac == 4 || miesiac == 5
|| dzien <= 20 && miesiac == 6) {
		printf("Wiosna");
	}
	else if (dzien >= 21 && miesiac == 6 || miesiac == 7 || miesiac
== 8 || dzien <= 20 && miesiac == 9) {
		printf("Lato");
	}
	else if (dzien >= 21 && miesiac == 9 || miesiac == 10 ||
miesiac == 11 || dzien <= 20 && miesiac == 12) {
		printf("Jesień");
	}
	else {
		printf("Zima");
	}
}
```

Program do obliczania pola figur z menu

```c
// Program oblicz. pole i obwód 3 figur (prostokąt, kwadrat, koło)
// które są wybierane za pomocą menu.
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main() {
	system("chcp 1250>nul");
	printf("Program do obliczania pola, wybierz liczbę: \n");
	printf("1. Prostokąt\n");
	printf("2. Kwadrat\n");
	printf("3. Koło\n");
	int menu;
	scanf("%d", &menu);

	switch (menu) {
	case 1: {
		float bok1, bok2;
		printf("Podaj długość boku 1: ");
		scanf("%f", &bok1);
		printf("Podaj długość boku 2: ");
		scanf("%f", &bok2);
		float pole = bok1 * bok2;
		float obwod = 2 * bok1 + 2 * bok2;
		printf("Prostokąt. Pole = %.2f, obwód = %.2f", pole, obwod);
		break;
	}
	case 2: {
		float bok;
		printf("Podaj długość boku kwadratu: ");
		scanf("%f", &bok);
		float pole = pow(bok, 2);
		float obwod = 4 * bok;
		printf("Kwadrat. Pole = %.2f, obwód = %.2f", pole, obwod);
		break;
	}
	case 3: {
		float r;
		printf("Podaj promień: ");
		scanf("%f", &r);
		float pole = M_PI * pow(r, 2);
		float obwod = 2 * M_PI * r;
		printf("Koło. Pole = %.2f, obwód = %.2f", pole, obwod);
		break;
	}
	default: printf("Wybrano niepoprawną opcję");
		break;
	}
}
```

Policzyć ile wśród 10 liczb wprowadzonych z klawiatury jest dodatnich, ujemnych, zerowych

```
#include <iostream>
using namespace std;

int main() {
	system("chcp 1250>nul");

	int liczba, dodatnie = 0, ujemne = 0, zerowe = 0;
	for (int i = 1; i <= 10; i++) {
		cout << "Wprowadź liczbę " << i << ": ";
		cin >> liczba;
		if (liczba > 0) {
			dodatnie += 1;
		}
		else if (liczba < 0) {
			ujemne += 1;
		}
		else {
			zerowe += 1;
		}
	}
	cout << "Liczb dodatnich: " << dodatnie << endl;
	cout << "Liczb ujemnych: " << ujemne << endl;
	cout << "Liczb zerowych: " << zerowe << endl;
}
```

Wczytać 10 liczb całkowitych. Obliczyć sumę liczb dodatnich i ujemnych.

```
#include <iostream>
using namespace std;

int main() {
     system("chcp 1250>nul");

     int liczba, sumaDodatnich = 0, sumaUjemnych = 0;
     for (int i = 1; i <= 10; i++) {
          cout << "Wprowadź liczbę " << i << ": ";
          cin >> liczba;
          if (liczba >= 0) {
               sumaDodatnich += liczba;
          }
          else {
               sumaUjemnych += liczba;
          }
     }
     cout << endl;
     cout << "Suma liczb dodatnich: " << sumaDodatnich << endl;
     cout << "Suma liczb ujemnych: " << sumaUjemnych << endl;
}
```

Obliczyć i wyświetlić jaką drogę pokona ciało spadające swobodnie po 1, 2, ..,10 sek.

```
#include <iostream>
#include <cmath>
using namespace std;

// S = a*pow(t,2)/2
const float a = 9.81;

int main() {
	system("chcp 1250>nul");
	float s = 0;
	for (int i = 1; i <= 10; i++) {
		s = a * pow(i, 2)/2;
		cout << "Droga po " << i << " s. wynosi: \t" << s << endl;
	}
}
```

Ile razy we wczytanym ciągu N liczb wystąpiła wartość z przedziału od 5 do 25 (włącznie)

```
#include <iostream>
using namespace std;

int main() {
	system("chcp 1250>nul");
	int n, liczba, iloscWystapien = 0;
	cout << "Ile liczb chcesz wprowadzić? : ";
	cin >> n;

	for (int i = 0; i < n; i++) {
		cout << "Podaj liczbę " << i + 1 << ": ";
		cin >> liczba;
		if (liczba >= 5 && liczba <= 25) {
			iloscWystapien += 1;
		}
	}
	cout << "Liczby w zakresie od 5 do 25 wystąpiły " <<
iloscWystapien << " razy.";
}
```

Obliczyć i wyświetlić kwadraty i sześciany N kolejnych liczb całkowitych

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
     system("chcp 1250>nul");
     int ileLiczb;
     cout << "Ile liczb ma być wyświetlonych? : ";
     cin >> ileLiczb;
     for (int i = 1; i <= ileLiczb; i++) {
          cout << i << " do potęgi 2 = " << pow(i, 2) << ", a do
potęgi 3 = " << pow(i, 3) << endl;
     }
}
```

Wczytaj ciąg liczb całkowitych zakończony zerem. Policz i wyświet ilość liczb dodatnich i ujemnych w ciągu

```
#include <iostream>
using namespace std;


int main(){
    system("chcp 1250>nul");
    int liczba, ileDodatnich = 0, ileUjemnych = 0;

    do {
        cout << "Podaj liczbę: ";
        cin >> liczba;
        if (liczba > 0) {
            ileDodatnich++;
        }
        else if(liczba < 0){
            ileUjemnych++;
        }
    } while (liczba != 0);

    cout << "Dodatnich: " << ileDodatnich << endl;
    cout << "Ujemnych: " << ileUjemnych << endl;
}
```

Sprawdź dla jakiego N suma szeregu $1^2+2^2+3^2+....+N^2< 10000$

```
#include <iostream>
#include <cmath>
using namespace std;

int main(){
    system("chcp 1250>nul");
    int suma = 0, n = 1;
    while (suma < 10000) {
        suma += pow(n,2);
        cout << "Dla n = " << n << ", suma ciągu wynosi: " <<
suma << endl;
        n++;
    }
    cout << "-----------------------------------------"
<< endl;
    cout << "Suma szeregu jest mniejsza od 10000 dla n = " <<
(n - 2);
}
```

Znajdź największą liczbę rzeczywistą, której trzecia potęga jest mniejsza od 2000.

```
#include <iostream>
#include <cmath>
using namespace std;

int main(){
    system("chcp 1250>nul");
    double potega;
    double n = 0.01;
    do {
        potega = pow(n, 3);
        cout << "Wartość 3ciej potęgi liczby " << n << " wynosi: "
<< potega << endl;
        n+=0.01;
    } while (potega < 2000);
    cout << "Największa liczba rzeczywista z dokł. do dwóch miejsc
po przecinku, której trzecia potęga jest mniejsza od 2000, wynosi: "
<< (n -= 0.02);
}
```

Znaleźć NWD (największy wspólny dzielnik) dwóch liczb całkowitych

```
#include <iostream>
#include <cmath>
using namespace std;

int main(){
    system("chcp 1250>nul");
    int dzielna,dzielnik,liczba1,liczba2,n=1,nwd = 1;
    cout << "Podaj liczbę 1: ";
    cin >> liczba1;
    cout << "Podaj liczbę 2: ";
    cin >> liczba2;

    if (liczba1 > liczba2) {
        dzielna = liczba1;
        dzielnik = liczba2;
    }
    else {
        dzielna = liczba2;
        dzielnik = liczba1;
    }

    do {
        if (dzielna % n == 0 && dzielnik % n == 0) {
            nwd = n;
        }
        n++;
    } while (n <= dzielnik);

    cout << "nwd liczb " << dzielna << " i " << dzielnik << "
wynosi: " << nwd;
}
```

NWD(największy wspólny dzielnik) i NWW (najmniejsza wspólna wielokrotność)

Program, który znajduje NWD(największy wspólny dzielnik) i NWW (najmniejsza wspólna wielokrotność) dwóch liczb całkowitych.

```
#include <iostream>
#include <cmath>
using namespace std;

void pomniejszLiczby(int* w_dzielna, int* w_dzielnik, int dz) {
    *w_dzielna /= dz;
    *w_dzielnik /= dz;
}

int main(){
    system("chcp 1250>nul");
    int dzielna, dzielnik, liczba1, liczba2, dzielnikStaly, n=2, nwd = 1, nww =
1;
    cout << "Podaj liczbę 1: ";
    cin >> liczba1;
    cout << "Podaj liczbę 2: ";
    cin >> liczba2;

    if (liczba1 > liczba2) {
        dzielna = liczba1;
        dzielnik = liczba2;
    }
    else {
        dzielna = liczba2;
        dzielnik = liczba1;
    }

    dzielnikStaly = dzielnik;

        do {
            if (dzielna % n == 0 && dzielnik % n == 0) {
                nwd *= n;
                pomniejszLiczby(&dzielna, &dzielnik, n);
            }
            else if(dzielna % n != 0 || dzielnik % n != 0 && n <= dzielnik){
                n++;
            }
            else {
                cout << "nwd: " << nwd << endl;
                cout << "nww: " << nwd * dzielna * dzielnik << endl;
                return 0;
            }
        } while (nwd<=dzielnikStaly);
}
```

NWD(największy wspólny dzielnik) i NWW (najmniejsza wspólna wielokrotność) bez wskaźników

```
#include <iostream>
using namespace std;


int main() {
	system("chcp 1250>nul");
	int liczba1, liczba2, dzielna, dzielnik, n=2, nwd = 1, nww = 1,
dzielnikPoczatkowy;
	cout << "Podaj liczbę 1:";
	cin >> liczba1;
	cout << "Podaj liczbę 2:";
	cin >> liczba2;

	if (liczba1 > liczba2) {
		dzielna = liczba1;
		dzielnik = liczba2;
	}
	else {
		dzielna = liczba2;
		dzielnik = liczba1;
	}
	dzielnikPoczatkowy = dzielnik;

	do {
		if (dzielna % n == 0 && dzielnik % n == 0) {
			nwd *= n;
			dzielna /= n;
			dzielnik /= n;
		}
		else if (dzielna % n != 0 || dzielnik % n != 0 && n <=
dzielnik) {
			n++;
		}
		else {
			cout << "nwd: " << nwd << endl;
			cout << "nww: " << nwd * dzielna * dzielnik << endl;
			return 0;
		}
	} while (nwd <= dzielnikPoczatkowy);
}
```

W tablicy n elementowej liczb całkowitych policz ilość elementów większych niż średnia wartość tych elementów.

```
/*
W tablicy N elementowej liczb całkowitych policzyć ilość elementów
większych niż
średnia wartość tych elementów.
*/
#include <iostream>
using namespace std;

int main() {
	system("chcp 1250>nul");
	const int n = 6;
	int suma = 0, iloscLiczbWiekszychOdSredniej = 0;
	float srednia;
	int tablica[n] = { 2,-3,4,8,1,5 };
	for (int i = 0; i < n; i++) {
		suma+=tablica[i];
	}
	srednia = suma / (float)n;
	for (int i = 0; i < n; i++) {
		if (tablica[i] > srednia) {
			iloscLiczbWiekszychOdSredniej++;
		}
	}
	cout << "Średnia: " << srednia << endl;
	cout << "Ilość elementów większych od średniej: " <<
iloscLiczbWiekszychOdSredniej;
}
```

Z n elementowej tablicy A do tablicy B skopiuj elementy większe od parametru k.

```
/*Napisać program, który z N elementowej tablicy A do tablicy B
kopiuje elementy większe od parametru k.*/

#include <iostream>
using namespace std;

int main() {
    system("chcp 1250>nul");
    int k = 6;
    const int n = 10;
    int tablica1[n] = { 1,2,6,4,5,6,2,8,1,10 };
    int tablica2[n];
    for (int i = 0; i < n; i++) {
        if (tablica1[i] > k) {
            tablica2[i] = tablica1[i];
        }
        else {
            tablica2[i] = 0;
        }
    }

    for (int i = 0; i < n; i++) {
        cout << tablica2[i] << ", ";
    }
}
```

W n elementowej tablicy losowych liczb całkowitych zamień wartości ujemne na ich kwadraty.

```
/*W n elementowej tablicy losowych liczb całkowitych zamień wartości
ujemne na ich kwadraty*/
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <cmath>
using namespace std;

int main() {
	srand(time(NULL));
	const int n = 10;
	int tablica[n];
	for (int i = 0; i < n; i++) {
		tablica[i] = rand() % 20 - 10;
		if (tablica[i] < 0) {
			tablica[i] = pow(tablica[i], 2);
		}
		cout << tablica[i] << ", ";
	}
}
```

Wyświetl indeksy i wartości elementów dodatnich tablicy.

```
/*Program, który wyświetla indeksy i wartości elementów dodatnich
tablicy.*/

#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main() {
     srand(time(NULL));
     const int n = 10;
     int tablica[n];
     for (int i = 0; i < n; i++) {
          tablica[i] = rand() % 50 - 25;
          if (tablica[i] > 0) {
               cout << i << ": " << tablica[i] << endl;
          }
     }
}
```

Liczba elementów tablicy o wartościach z zakresu <p;q>

```
/*Napisać program , który oblicza liczbę elementów tablicy o
wartościach z przedziału p-q*/

#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main() {
	srand(time(NULL));
	const int n = 10;
	int p = 2, q = 6, liczbaElementow = 0;
	int tablica[n];
	for (int i = 0; i < n; i++) {
		tablica[i] = rand() % 21;
		if (tablica[i] > p && tablica[i] < q) {
			liczbaElementow++;
		}
	}

	for (int i = 0; i < n; i++) {
		cout << tablica[i] << ", ";
	}
	cout << endl<<"Liczba elementow wynosi: " << liczbaElementow;
}
```

Punkty na płaszczyźnie wewnątrz okręgu

```
/*
Utworzyć tablicę z informacjami o 10 punktach losowych na
płaszczyźnie (współrzędne x i y).
Wyświetlić dane punktów, które znajdują się w okręgu o środku w
punkcie (5,5) i promieniu 3.
*/

#include <iostream>
#include <cstdlib>
#include <cmath>
#include <ctime>
using namespace std;
int main() {
	srand(time(NULL));
	int tablica[10][2];
	int srX=6, srY=6, r = 5;
	for (int i = 0; i < 10; i++) {
		tablica[i][0] = rand() % 11;
		tablica[i][1] = rand() % 11;
		if (pow(tablica[i][0] - srX, 2) + pow(tablica[i][1]-srY,
2) < pow(r,2)) {
			cout << "punkt x= " << tablica[i][0] << ", y=" <<
tablica[i][1] << " znajduje sie w srodku okregu \n";
		}
	}
}
```

Suma każdego wiersza i kolumny w macierzy 6 x 5

```
/* Wprowadź(wylosuj) liczby do dwuwymiarowej tablicy liczb
całkowitych o wymiarach
5 wierszy na 6 kolumn oraz oblicz sumę wyrazów w wierszach i w
kolumnach.
Wyświetl zawartość tablicy oraz obliczone sumy z numerem wiersza i
kolumny.*/

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
     srand(time(NULL));
     const int w = 5, k = 6;
     int tablica[w][k], sumaw = 0, sumak = 0;
     for (int i = 0; i < w; i++) {
          for (int j = 0; j < k; j++) {
               tablica[i][j] = rand() % 10;
               cout << tablica[i][j] << " ";
          }
          cout << endl;
     }

     for (int i = 0; i < w; i++) {
          sumaw = 0;
          for (int j = 0; j < k; j++) {
               sumaw += tablica[i][j];
          }
          cout << "suma wiersza " << i << " wynosi " << sumaw <<
endl;
     }

     for (int i = 0; i < k; i++) {
          sumak = 0;
          for (int j = 0; j < w; j++) {
               sumak += tablica[j][i];
          }
          cout << "suma kolumny " << i << " wynosi " << sumak <<
endl;
     }
}
```

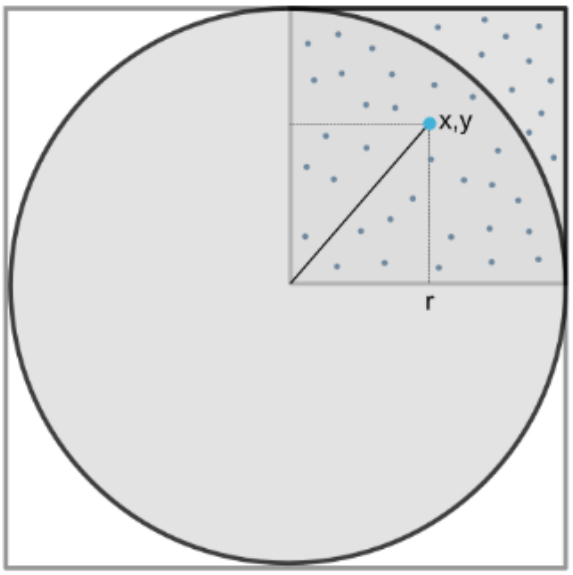
Numer wiersza o największej sumie elementów

```
/* W tablicy dwuwymiarowej znaleźć numer wiersza o największej sumie
elementów. */

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
	srand(time(NULL));
	const int w = 4, k = 4;
	int tablica[w][k], sumaEl = 0, sumaMaks = 0, najwiekszyNrWiersza =
0;

	for (int i = 0; i < w; i++) {
		for (int j = 0; j < k; j++) {
			tablica[i][j] = rand() % 10;
			cout << tablica[i][j] << " ";
		}
		cout << endl;
	}

	for (int i = 0; i < w; i++) {
		for (int j = 0; j < k; j++) {
			sumaEl += tablica[i][j];
		}
		if (sumaEl > sumaMaks) {
			sumaMaks = sumaEl;
			najwiekszyNrWiersza = i;
		}
		cout << i << ": " << sumaEl << endl;
		sumaEl = 0;
	}
	cout << "Najwieksza suma el. jest w wierszu o indeksie: " <<
najwiekszyNrWiersza;
}
```

Usuwanie wskazanego wiersza z tablicy dwuwymiarowej

```
/* Napisać program usuwania zawartości wskazanego wiersza z tablicy dwuwymiarowej(z
kompresją pozostałej zawartości tablicy – ostatni wiersz wypełnić wartością 0). */

#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

const int wiersze = 5;
const int kolumny = 5;

int main() {
	system("chcp 1250>nul");
	srand((unsigned)time(NULL));
	int tab1[wiersze][kolumny], tab2[wiersze][kolumny], wierszDoUsuniecia;

	for (int i = 0; i < wiersze; i++) {
		for (int j = 0; j < kolumny; j++) {
			tab1[i][j] = rand() % 10;
			cout << tab1[i][j] << " ";
		}
		cout << endl;
	}
	do {
		cout << "Który wiersz usunąć (Podaj numer indeksu od 0 do " << wiersze -
1<< " ): ";
		cin >> wierszDoUsuniecia;
	} while (wierszDoUsuniecia <0 || wierszDoUsuniecia > wiersze -1);

	for (int i = 0; i < wiersze;i++) {
		for (int j = 0; j < kolumny; j++) {
			if (i < wierszDoUsuniecia) {
				tab2[i][j] = tab1[i][j];
			}
			else {
				if (i < wiersze - 1) {
					tab2[i][j] = tab1[i+1][j];
				}
				else {
					tab2[i][j] = 0;
				}
			}
		}
	}

	for (int i = 0; i < wiersze; i++) {
		for (int j = 0; j < kolumny; j++) {
			cout << tab2[i][j] << " ";
		}
		cout << endl;
	}
}
```

Wyznaczanie liczby pi za pomocą losowych punktów wewnątrz okręgu



$$\frac{\pi r^2}{(2 \cdot r)^2} = \frac{\text{p.w okręgu}}{\text{p.w kwadracie}}$$

$$\pi = 4 \cdot \text{pwo} / \text{pwk}$$

```
/* Obliczanie liczby pi za pomocą losowych punktów wewnątrz okręgu i kwadratu
Liczbe Pi można obliczyć dzieląc pole okręgu o promieniu r przez pole kwadratu
opisanego na tym okręgu. Na koniec należy wynik pomnożyć przez 4.*/

#include <iostream>
#include <ctime>
using namespace std;

float generujLosowyPunkt(int n) {
	int ilPunktowWOkregu = 0, ilPunktowWKwadracie = 0;
	for (int i = 0; i < n; i++) {
		float x = rand() % 101 / 100.0;
		float y = rand() % 101 / 100.0;
		float odleglosc = x * x + y * y;
		if (odleglosc <= 1) {
			ilPunktowWOkregu++;
		}
		ilPunktowWKwadracie++;
	}
	return (float)4 * ilPunktowWOkregu / ilPunktowWKwadracie;
}

int main() {
	srand((unsigned)time(NULL));
	cout << generujLosowyPunkt(10000);
}
```

Suma cyfr podanej liczby

```
// Znajdź sumę cyfr podanej liczby
#include <iostream>
using namespace std;

int main() {
	system("chcp 1250>nul");
	int num, suma = 0;
	cout << "Podaj liczbę: ";
	cin >> num;
	while (num > 0) {
		suma += num % 10;
		num /= 10;
	}
	cout << endl << "Suma cyfr wynosi " << suma;
}
```

Wypisanie ciągu Fibonacciego do n-tego elementu

```
// Wypisanie ciągu fibonacciego do n-tego elementu (0 1 1 2 3 5
8 13 21 34 ...)
#include <iostream>
using namespace std;

int main() {
     system("chcp 1250>nul");
     int num1 = 0, num2 = 1, n, numNext;
     cout << "Podaj n-ty wyraz ciągu: ";
     cin >> n;
     if (n <= 0) {
          return 0;
     }
     else {
          for (int i = 0; i < n; i++) {
               if (i == 0) {
                    cout << num1 << ", ";
               }
               else if (i == 1) {
                    cout << num2 << ", ";
               }
               else {
                    numNext = num1 + num2;
                    num1 = num2;
                    num2 = numNext;
                    cout << numNext << ", ";
               }
          }
     }
}
```

Struktura, książki

Proszę utworzyć tablicę struktur z informacjami o książkach:

nazwisko autora

tytuł

liczba wypożyczeń w roku akademickim

Wyświetlić listę wszystkich książek oraz tych, które nie były wypożyczane

```
#include <iostream>
using namespace std;

struct ksiazka {
	string nazwiskoAutora;
	string tytul;
	int liczbaWypozyczen;
};
int main() {
	system("chcp 1250>nul");
	ksiazka ksiazki[] = {
		{"Tolkien", "Władca pierścieni", 2},
		{"Brown", "Cyfrowa twierdza", 4},
		{"Sapkowski", "Krew elfów", 0},
		{"Rowling", "Harry Potter",0},
		{"Canavan", "The High Lord",1}
	};

	for (int i = 0; i < 5; i++) {
		cout << ksiazki[i].nazwiskoAutora << ", ";
		cout << ksiazki[i].tytul << ", ";
		cout << "Liczba wypożyczeń: " << ksiazki[i].liczbaWypozyczen <<
endl << endl;
	}
	cout << endl << "Książki, które nie były wypożyczone: " << endl;

	for (int i = 0; i < 5; i++) {
		if (ksiazki[i].liczbaWypozyczen == 0) {
			cout << ksiazki[i].nazwiskoAutora << ", ";
			cout << ksiazki[i].tytul << endl;
		}
	}
}
```

Struktura, książki z cin

```
/*Proszę utworzyć tablicę struktur  z informacjami o książkach :
nazwisko autora, tytuł, liczba wypożyczeń w roku akademickim
Wyświetlić listę wszystkich książek oraz tych, które nie były wypożyczane
*/

#include <iostream>
#include <string>
using namespace std;

struct ksiazka {
	string nazwiskoAutora;
	string tytul;
	int liczbaWypozyczen;
};

int main() {
	system("chcp 1250>nul");
	const int n = 3;
	ksiazka ksiazki[n];

	for (int i = 0; i < n; i++) {
		cout << "Podaj autora: ";
		getline(cin, ksiazki[i].nazwiskoAutora);
		cout << "Podaj tytuł: ";
		getline(cin, ksiazki[i].tytul);
		cout << "Podaj ilość wypożyczeń: ";
		cin >> ksiazki[i].liczbaWypozyczen;
		cin.ignore();
	}
	cout << endl;

	for (int i = 0; i < n; i++) {
		cout << ksiazki[i].nazwiskoAutora << ", ";
		cout << "\"" << ksiazki[i].tytul << "\"" << ", ";
		cout << "Liczba wypożyczeń: " << ksiazki[i].liczbaWypozyczen <<
endl;
	}

	cout << endl << "Książki, które nie były wypożyczone: " << endl;
	for (int i = 0; i < n; i++) {
		if (ksiazki[i].liczbaWypozyczen == 0) {
			cout << ksiazki[i].nazwiskoAutora << ", ";
			cout << "\"" << ksiazki[i].tytul << "\"" << endl;
		}
	}
}
```

Struktura, książki z cin

Proszę utworzyć tablicę struktur z informacjami o turnusach wczasowych:

kraj, liczba dni, cena na osobę, liczba miejsc

Wyświetlić pełną ofertę biura podróży oraz listę wczasów cenie powyżej 1000 zł.

```
#include <iostream>
using namespace std;

const int n = 3;
struct wycieczka {
	string kraj;
	int liczbaDni;
	int cena;
	int liczbaMiejsc;
} wycieczki[n];

int main() {
	system("chcp 1250>nul");

	for (int i = 0; i < n; i++) {
		cout << "Podaj kraj: ";
		cin >> wycieczki[i].kraj;
		cout << "Podaj ilosc dni: ";
		cin >> wycieczki[i].liczbaDni;
		cout << "Podaj cene: ";
		cin >> wycieczki[i].cena;
		cout << "Podaj liczbe miejsc: ";
		cin >> wycieczki[i].liczbaMiejsc;
	}

	for (int i = 0; i < n; i++) {
		cout << wycieczki[i].kraj << ", ";
		cout << "ilość dni: " << wycieczki[i].liczbaDni << ", ";
		cout << "cena: " << wycieczki[i].cena << ", ";
		cout << "Liczba miejsc: " << wycieczki[i].liczbaMiejsc << endl;
	}
	cout << endl << "Lista wczasow powyzej 1000zl: " << endl;

	for (int i = 0; i < n; i++) {
		if (wycieczki[i].cena > 1000) {
			cout << wycieczki[i].kraj << ", ";
			cout << "ilość dni: " << wycieczki[i].liczbaDni << ", ";
			cout << "cena: " << wycieczki[i].cena << ", ";
			cout << "Liczba miejsc: " << wycieczki[i].liczbaMiejsc <<
endl;
		}
	}
}
```