SKŁADNIA INSTRUKCJI IF ORAZ SWITCH

```
if (wyrażenie_1)
  instrukcja_1;
else if (wyrażenie_2)
  instrukcja_2;
else if (wyrażenie_3)
  instrukcja_3;
else
  instrukcja_4;
```

```
switch (wyrażenie_całkowite)
{
case wartość_1:
    instrukcja_1;
    break;
case wartość_2:
case wartość_3:
case wartość_4:
    instrukcja_234;
    break;
default:
    instrukcja_domyslna;
    break;
}
```

Przykład:

```
int a;
cin >> a;
switch (a)
{
case 1:
    cout << "Ala";
    break;
case 2:
    cout << "Ola";
case 3:
case 4:
    cout << "Ula";
    break;
case 6:
    cout << "Ela";
default:
    cout << "Agata";
    break;
}
```

WYRAŻENIE A INSTRUKCJA

Wyrażenie:

- nazwa zmiennej,
- wywołanie funkcji,
- nazwa tablicy,
- stała,
- nazwa funkcji,
- odwołanie do elementu struktury,
- odwołanie do elementu tablicy,
- jedna z powyższych postaci z nawiasami i/lub operatorami.

Wyrażenie zakończone średnikiem jest **instrukcją**

```
int a = 0, b = 2;

if ( a         )   cout << "Ala";
if ( a == 0    )   b = 7;
if ( !(a == 0))   b;
if ( !a        )   7;
if ( 123       )   ;
if ( a && b    )   a == 0;
if ( cos(90)   )   { cout << "Ola";  a = 10; b = (a == 5); }
```

SKŁADNIA PĘTLI WHILE

```
while (wyrażenie)
  instrukcja;
```

```
while (wyrażenie)
{
    instrukcja_1;
    instrukcja_2;
    . . .
    instrukcja_N;
}
```

Przykład:

```
int w = 15, ile;
cout << "W ramach promocji restauracji \"AMAM\" mamy do "
        "rozdania 15 smacznych bulek.\nRozdajemy je "
        "nadjezdzajacym w samochodach ludziom.\n\n";

while (w>0)
{
   cout << "Ile mam dac bulek?";
   cin >> ile;
   if (ile > w) cout << "Zostaly mi tylko " << w << '.';
   cout << "Prosze \n";
   w = w - ile;
}
cout << "Przykro mi. Dzisiejsze bulki się już skonczyly.";
```

Pętla wykonywana jest tak długo jak wartość **wyrażenie** jest różna od zera

PREZENTACJA KODÓW ASCI

```
char litera;
cout << "Podaj litere: ";
cin >> litera;
int kodAsci = int(litera);
cout << "Wcisnales litere \'" << litera<< "\',\n"<<
    "ktorego kod ASCI wynosi " << kodAsci << endl;
system("pause");
```

```
Podaj litere: k
Wcisnales litere 'k',
ktorego kod ASCI wynosi 107
Aby kontynuować, naciśnij dowolny klawisz . . .
```

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

int main()
{
	char znak = ' ';
	cout << "Kody znaków ASCI\n";
	cout << "ESC - wyjście\n\n";
	while (znak != 27)
	{
		znak = getch();
		cout << znak << " - " << int(znak) << endl;
	}
}
```

```
Kody znakow ASCI
ESC - wyjscie

a - 97
b - 98
A - 65
B - 66
  - 32
/ - 47
? - 63
{ - 123
} - 125
[ - 91
] - 93
```

PĘTLA WHILE

```
int i, n = 5;
cout << "Wyswietlanie n kolejnych liczb";
cout<< endl << endl;

i = 1;
while (i <= n)
{
    cout << "Liczba " << i << endl;
    i++;
}
```

```
int i, n = 5;
cout << "Wyswietlanie n kolejnych liczb";
cout<< endl << endl;

i = 0;
while (i < n)
{
    cout << "Liczba " << i+1 << endl;
    i++;
}
```

SKŁADNIA PĘTLI FOR

for(wyrażenie_inicjujace ; wyrażenie_testujace ; wyrażenie_modyfikujace)
{
   wykonywana_instrukcja ;
}

jest równoważna konstrukcji:

wyrazenie_inicjujace **;**
while(*wyrazenie_testujace* **)**
{
  wykonywana_instrukcja ;
  wyrazenie_modyfikujace ;
}

Przykład z porównaniem:

```
int i, n = 5;

i = 0;
while (i < n)
{
    cout << "Liczba " << i+1 << endl;
    i++;
}
```

```
int i, n = 5;

for (i = 0; i < n; i++)
{
    cout << "Liczba " << i+1 << endl;
}
```

ŁAMIGŁÓWKA - PĘTLA FOR

```
for (int i = 2; i <= 8; i += 2)
    cout << "Ala" << endl;
```
// 4 razy

```
for (int i = 9; i>0; i = i - 3)
    cout << "Ola" << endl;
```
// 3 razy

```
for (int i = 5; i >= 0; i--)
    cout << "Iza" << endl;
```
// 6 razy

```
for (int i = 18; i>4; i--);
    cout << "Ula" << endl;
```
// 1 raz

```
for (int i = 2; i == 7; i = i + 1)
    cout << "Ela" << endl;
```
// 0 razy

```
for (int i = 7; i<0; i = i - 2)
    cout << "Ewa" << endl;
```
// 0 razy

```
for (int i = 0; i<12; i + 2)
    cout << "Ana" << endl;
```
// nan razy

```
for (int i=1, j=0; j<9; i++,j+=i)
    cout << "Aga" << endl;
```
// 3 razy

SKŁADNIA PĘTLI DO{…}WHILE

Pętla ta jest wykonywana co najmniej jeden raz.

```
do
   instrukcja;
while ( wyrażenie );
```

```
do
{
   instrukcja_1;
   instrukcja_2;
    . . .
   instrukcja_N;
}
while ( wyrażenie );
```

Przykłady:

```
//wymuszenie pobrania wartości nieujemnej
int a;
do
{
    cout << "Podaj liczbę dodatnia a=";
    cin >> a;
}   while (a<0);
```

```
//cykliczne wykonywanie programu

char znak;
do
{
    // instrukcje programu
    // ...
    cout << "Czy chcesz zakonczyc"
            " program (t/n)?";
    cin >> znak;
} while (znak != 't');
    cout << endl << "Koniec programu ";
```

INSTRUKCJE BREAK I CONTINUE W RÓŻNEGO TYPU PĘTLACH

break;

jest używana do natychmiastowego opuszczenia pętli while, for, do...while lub instrukcji switch

continue;

służy do wcześniejszego zakończenia kolejnej iteracji pętli for, do...while lub while; wykonanie pętli jest kontynuowane

```
while( warunek)
{
  ...
  break;
  ...
  continue;
 ...
}
...
```

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

void main(void)
{
    int losowa;
    srand(time(NULL));
    cout << "Ciag liczb niezerowych\n";

    while (1)
    {
        losowa = rand() % 10;
        if (losowa == 0) break;
        cout << losowa << ' ';
    }
}
```

```
Ciag liczb niezerowych
5 6 9 5 2 9
```

```
// Wariacja bez powtorzen
int n = 4;
for (int i = 1; i <= n; i++)
{
    for (int j = 1; j <= n; j++)
    {
        if (i == j)
            continue;
        cout << i << "," << j << endl;
    }
}
```

```
1,2
1,3
1,4
2,1
2,3
2,4
3,1
3,2
3,4
4,1
4,2
4,3
```

INSTRUKCJA GOTO

Ostatecznie można ją zastosować tam, gdzie doprowadzi do istotnego uproszczenia zawikłanej struktury programu. Kod programu staje się przez nią mało czytelny.

...
goto *etykieta*;
...
...
etykieta:
...
...

```
goto  etykieta;
cout << "Ala";
etykieta:
cout << "Ola";
```

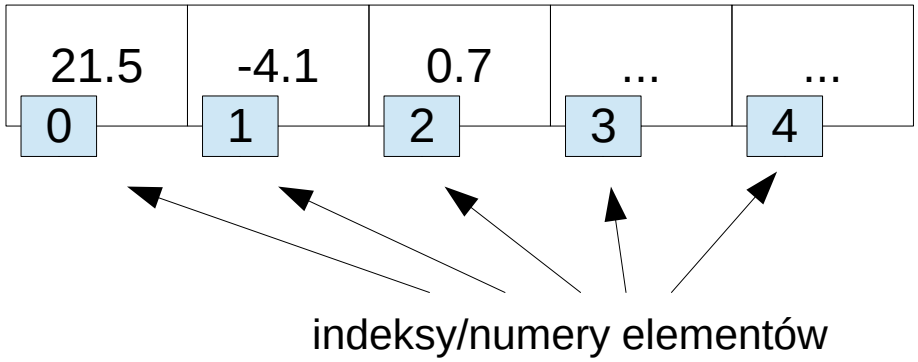
STAŁE WYLICZENIOWE

```
enum dzien {
    niedziela, poniedzialek, wtorek,
    sroda, czwartek, piatek, sobota
};

dzien wyklad_z_inf ;
dzien wyklad_z_fiz = poniedzialek;

wyklad_z_inf = sroda;

cout << "W srode mamy wyklady z:\n";
if (wyklad_z_inf == sroda)  cout << " - informatyki\n";
if (wyklad_z_fiz == sroda)  cout << " - fizyki\n";
// itd.
```

```
enum kolor
{
    pomaranczowy, zolty, czerwony = 11, zielony = 12, niebieski = 15,
    czarny, szary, fioletowy, red = 11, green = 12, blue = 14
} tlo = czarny, pioro;

pioro = green;
cout << tlo;
if (pioro == zielony)
    cout << "Zielone pioro";
```

TABLICE

```
// double t0,t1,t2,t3,t4;
double t[5];
t[0] = 21.5;
t[1] = -4.1;
t[2] = 0.7;
```



Ogólna postać definicji tablicy:

typ_elementu nazwa_tablicy [liczba_elementów] ;

Tablica - jest reprezentacją umożliwiającą zgrupowanie kilku danych tego samego typu i odwoływanie się do nich za pomocą wspólnej nazwy. Jest to jeden z najczęściej wykorzystywanych typów danych.

Przykłady:

```
double t[5];          // 5-cio elementowa tablica liczb rzeczywistych
int tab[10];          // 10-cio elementowa tablica liczb całkowitych
char tekst[255];      // 255-cio elementowa tablica znaków
double(*funkcje[20]) (double, double); // tablica 20 wskaźników na funkcje
```

Uwagi o tablicach:

- spójny obszar pamięci
- równomierne rozmieszczenie kolejnych elementów bezpośrednio jeden po drugim.
- elementy tablicy są indeksowane od zera !
- nie jest sprawdzana poprawność (zakres) indeksów!
- zwykła tablica nie przechowuje informacji o liczbie swoich elementów

```
int tab[5] = { 2, 3, 4, 7, 6 };
int a = 3;

cout << tab[1];                    // 3
cout << tab[5]; //tab[5]=0; // -163721143
tab[2] = tab[4];                   // tab[2] z 4 na 6
tab[a] = a + tab[a + 1];           // tab[3]=3+6
cout << tab;                       // 7e8a3755
```

TABLICE ZNAKÓW A ŁAŃCUCHY

```
char a = 'x';
int  tabI[10] = { 20, -3, 12, -1, 0 };
char tabA[5] = { a, 'b', 'f', 'g', 'd' };
char tabB[5] = { 'e', '\n', 27 };
char tabC[] = { 'f', '\t', 27, ' ' };
char tabD[5] = { 'A', 'n', 'i', 'a' };
char tabE[10] = "Ela";
char tabF[10] = { 'E', 'l', 'a', 0 };
char tabG[] = { "Afrodyta" };

cout << tabE[1];      // I
cout << tabE;         // Ela
cout << tabD;         // Ania●#●&G@ăćėěĴ l
tabE[1] = 'w';
cout << tabE;         // Ewa
tabG[4] = 0;
cout << tabG;         // Afro
```

OPERACJE NA ŁAŃCUCHACH

```
int rozmiar;
char nazwa[20] = "Anatomia";
char n1[20], n2[20];

cout << sizeof(rozmiar);  // 4
cout << sizeof(int);      // 4
cout << sizeof(nazwa);    // 20
rozmiar = strlen(nazwa);
cout << rozmiar;          // 8
strcpy(n1, nazwa);
cout << n1;               // Anatomia
strncpy(n2, nazwa, 3);    n2[3] = 0;
cout << n2;               // Ana
if (strcmp(n1, n2)) cout << "Lancuchy sa rozne";
else                      cout << "Lancuchy sa takie same";
strcat(n1, n2);                                  // Lancuchy sa rozne
cout << n1;    // AnatomiaAna
cout << n2;    // Ana
```

WKORZYSTANIE STAŁYCH DO DEFINIOWANIA ILOŚCI ELEMENTÓW TABLICY

1. Rozmiar zadany bezpośrednio

```
int tablica[100];
```

2. Definicja stałej w stylu języka C

```
#define ROZMIAR 100
int tablica[ROZMIAR];
```

3. Definicja stałej w stylu języka C++

```
const int ROZMIAR_2 = 100;
int tablica_2[ROZMIAR_2];

-przykład dalszego wykorzystania stałej
for (int i = 0; i < ROZMIAR; i++)
  cout << tablica[i] << endl;
```

ZASTOSOWANIE INSTRUKCJI FOR DO OPERACJI NA TABLICACH

```
int i;
const int ROZMIAR = 10;

//definicja tablicy liczb rzeczywistych
double tablica[ROZMIAR];

//przypisanie kolejnymi liczbami parzystymi: 0,2,4,6 ,...
for (i = 0; i<ROZMIAR; i++)
     tablica[i] = 2 * i;

//wczytanie zawartości elementów tablicy z klawiatury
for (i = 0; i<ROZMIAR; i++)
{
     cout << "Podaj Tab["<<i+1<<"] = ";
     cin >> tablica[i];
}

// wyświetlenie zawartości elementów tablicy
for (i = 0; i<ROZMIAR; i++)
     cout << "Tab[" << i + 1 << "] = " << tablica[i]<<endl;

//zsumowanie wartości elementów tablicy
double suma = 0;
for (i = 0; i<ROZMIAR; i++)
     suma = suma + tablica[i];   //suma+=tablica[i];
cout <<endl<< "Suma = " << suma << endl;

//zliczenie ilości elementów o dodatnich wartościach
int ilosc = 0;
for (i = 0; i<ROZMIAR; i++)
     if (tablica[i]>0)
          ilosc = ilosc + 1;              // ilość += 1; lub ilość++;
cout << "Ilość dodatnich elementów = " << ilosc << endl;
```

```
Podaj Tab[1] = 6
Podaj Tab[2] = 7
Podaj Tab[3] = 8
Podaj Tab[4] = 9
Podaj Tab[5] = -2
Podaj Tab[6] = -3
Podaj Tab[7] = -4
Podaj Tab[8] = -5
Podaj Tab[9] = -6
Podaj Tab[10] = 10
Tab[1] = 6
Tab[2] = 7
Tab[3] = 8
Tab[4] = 9
Tab[5] = -2
Tab[6] = -3
Tab[7] = -4
Tab[8] = -5
Tab[9] = -6
Tab[10] = 10

Suma = 20
Ilosc dodatnich elementow = 5
```

TABLICE WIELOWYMIAROWE

Definicja tablicy wielowymiarowej:
typ_elementu nazwa_tablicy [wymiar_1] [wymiar_2] [wymiar_3] . . . ;

```
double tensor[4][4][4];    //tensor 3-go rzędu

double m[5][2];    //macierz; 5 wierszy po 2 kolumny
```

m[0][0]	m[0][1]
m[1][0]	m[1][1]
m[2][0]	m[2][1]
m[3][0]	m[3][1]
m[4][0]	m[4][1]

Reprezentacja tej macierzy w pamięci komputera

m[0][0]	m[0][1]	m[1][0]	m[1][1]	m[2][0]	m[2][1]	m[3][0]	m[3][1]	m[4][0]	m[4][1]

```
int Ala[5][2] = { { 2, 7 }, { 3, 9 }, { 4, 12 }, { 11, 22 }, { 1, 1 } };
int Ola[5][2] = { 2, 7, 3, 9, 4, 12, 11, 22, 1, 1 };
int Ela[5][2] = { { 2, 7 }, 3, 9, 4, 11 };

cout << Ala[3][0]; // 11
cout << Ola[1][1]; // 9                                          // 11
cout << Ela[4][1]; // 0
cout << Ela[2][1]; // 11                                         // 11
cout << Ala[0][2]; // 3

        char Imie[5][20] = { "Alicja", "Ola", "Natalia", "Adam" };
        cout << Imie[1];          // Ola
        cout << Imie[5];          // ●#●&G@ăćėěĴl
        cout << Imie[4];          //
        Imie[1][0] = 'E';
        cout << Imie[1];          // Ela
        strcpy(Imie[1], Imie[2]);
        cout << Imie[1];          // Natalia
```

STAŁE

typ	składnia	przykład	
char	ujęte w apostrofy	'a'	
ciąg znaków	ujęte w cudzysłowy	"abc"	
int	bez zera na początku	17	
l.ósemkowa		015	
l.szesnastkowa		0x2f	
long int		123l	123L
float		3.24f	3.2E7F
double		3.2e7	3.2E7
long double		3.2e7l	3.2E7L

```
char z1 = 'a', z2 = 98, z3 = '\"';
cout << z1 << ' ' << z2 << ' ' << z3 << endl;
```

```
a b "
```

```
int a = 23, b = 023, c = 0x23;
cout << a << ' ' << b << ' ' << c << endl;
```

```
23 19 35
```

WEJŚCIE/WYJŚCIE W C++ - FLAGI

W pliku **iostream.h** zdefiniowana jest zmienna wyliczeniowa (enum) zawierająca następujące flagi:

skipws – ignoruj białe znaki
left right internal - wyrównaj do lewej, prawej, do środka szerokości
dec oct hex – konwersja dziesiętna, ósemkowa, szesnastkowa
showbase - pokaż podstawę konwersji
showpoint - pokaż kropkę dziesiętną
showpos - znak + w liczbach dodatnich
uppercase - wielkie litery (w liczbach)
scientific fixed - notacja naukowa, zwykła
boolalpha - we/wy danych logicznych

Funkcje (metody)

setf(…) **unsetf**(…)	- włączenie / wyłączenie flag
width(int w)	- ustawienie szerokości pola
fill(int ch)	- wypełnianie pustych miejsc pola znakiem ch
precision(int p)	- ilość miejsc po kropce dziesiętnej

WEJŚCIE/WYJŚCIE W C++ - SPOSÓB UŻYCIA FLAG

```
long flaga1 = cout.flags();

int a=-12;  // JUSTOWANIE, width - ustawienie szerokości pola
cout<<"|"; cout.width(8); cout<<left    << a <<"|"<<endl;
cout<<"|"; cout.width(8); cout<<right   << a <<"|"<<endl;
cout<<"|"; cout.width(8); cout<<internal<< a <<"|"<<endl;
```

```
|-12     |
|     -12|
|-     12|
```

```
int b=33;   // SYSTEM WYSWIETLANIA: dec(10), oct(8), hex(16)
cout.setf(ios::hex);
cout.unsetf(ios::dec);                cout<< b <<endl;
cout.setf(ios::oct, ios::basefield); cout<< b <<endl;
cout.setf(ios::dec, ios::basefield); cout<< b <<endl;
```

```
21
41
33
```

```
cout.flags(flaga1);
int c=59;  cout.setf(ios::hex, ios::basefield);
cout.setf(ios::showpos);   cout<<c<<endl;  // dodaje + przed liczbami
cout.setf(ios::uppercase); cout<<c<<endl;  // powiększa litery (zapis 16-stkowy)
cout.setf(ios::showbase);  cout<<c<<endl;  // pokazuje podstawę konwersji
```

```
3b
3B
0X3B
```

```
cout.flags(flaga1);
float x =1175;
cout<<x<<endl;
cout.setf(ios::showpoint);  cout<<x<<endl; // wypisuje kropkę dziesietną
cout.setf(ios::scientific); cout<<x<<endl; // notacja naukowa (wykładnicza)
cout.setf(ios::fixed);      cout<<x<<endl; // notacja zwykła
```

```
1175
1175.00
1.175000e+03
1175.00
```

```
double tab2[]={1.23456789012, 2.3456, 3.2};
cout.precision(10);  //okresla dokladność wypisywanych liczb (domniemanie:6)
for(int i=0;i<3;i++)
{
  cout<< "| ";  cout.width( 2);  cout.fill('*'); cout<<i+8;
  cout<<" | ";  cout.width(14);  cout.fill('#'); cout<<tab2[i]<<" |\n";
}
```

```
| *8 | ###1.234567890 |
| *9 | ###2.345600000 |
| 10 | ###3.200000000 |
```

WEJŚCIE/WYJŚCIE W C++ - FUNKCJE I MANIPULATORY

Manipulatory włącza plik nagłówkowy **iomanip.h**

```
#include <iomanip>
```

dec hex oct	- konwersja
setioflags(long flaga)	- ustawianie
resetioflags(long flaga)	cofanie flag wg bitów w zmiennej flaga
setbase (int base)	- ustawienie podstawy
setw(int w)	- ustawienie szerokości pola
setfill(int ch)	- wypełnianie pustych miejsc pola znakiem ch
setprecision(int p)	- ilość miejsc po kropce dziesiętnej
ws	- pomiń początkowe spacje
endl ends	- koniec linii, koniec łańcucha(znak pusty)
flush	- wyczyszczenie strumienia

WEJŚCIE/WYJŚCIE W C++ - FUNKCJE I MANIPULATORY DO FORMATOWANIA

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

int main(void)
{
 cout.setf(ios::hex);
 cout.unsetf(ios::dec);
 cout << 127 << endl;
 cout << 128 << endl;
 cout.setf(ios::dec);
 cout.unsetf(ios::hex);

 cout << endl;
 cout.fill('*');
 for (int i = 10; i<100000; i *= 10)
 {
    cout << "suma: ";
    cout.width(9);
    cout << i << " zl\n";
 }

 cout << endl;
 cout.setf(ios::fixed);
 cout.precision(9);
 cout << 12.34 << endl;
}
```

```
7f
80
```

```
suma: ********10 zl
suma: *******100 zl
suma: ******1000 zl
suma: *****10000 zl
```

```
12.340000000
```

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

int main(void)
{
 cout << hex;
 cout << 127 << endl;
 cout << 128 << endl;
 cout << setbase(10);  // cout<<dec;

 cout << endl << setfill('*');
 for (int i = 10; i<100000; i *= 10)
    cout << "suma: "<<setw(9)<<i<<" zl\n";

cout << endl << setiosflags(ios::fixed)
    << setprecision(9);
cout << 12.34 << endl;
}
```

FUNKCJE WŁASNE

Składnia definicji funkcji

zwracany_typ NAZWA_FUNKCJI(lista_parametrów)
{
  instrukcja lub sekwencja instrukcji;
}

```
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
using namespace std;

int main(void)
{
    double alfa = 0.1*M_PI;
    double x, y, z, u;

    x = sin(alfa);
    y = tan(2*alfa)*sqrt(1.0/3.0);
    z = 5.0*cos(3.0)*pow(0.45, 2.0/3);
    // …
}
```

```
#include <iostream>
using namespace std;
// ------------------------------------
double srednia(int a, int b)
{
    double s;
    s = (a + b) / 2.0;
    return s;
}
// ------------------------------------
int main(void)
{
    int x, y, w;
    cout << "Podaj dwie liczby";
    cin >> x >> y;

    w = srednia(x, y); // w = pow(x,y);
    cout << "srednia=" << w;

    w = srednia(2, y);
    cout << "inna srednia=" << w;
}
```

FUNKCJA, CD

- lista parametrów może być pusta
  lub zawierać opisy kolejnych parametrów (pooddzielane przecinkami):

  main( ) main(**void**) main(**int** argc , **char*** argv[ ])

- parametry definiowane są tak jak zmienne.
  Uwaga: nie można grupować sekwencji parametrów tego samego typu:

  int srednia (**int** liczba_1, liczba_2, liczba_3) ← źle !

- „ciało” funkcji jest zawarte pomiędzy nawiasami: **{ ... }**
  (bez średnika na końcu)

- działanie funkcji kończy się po napotkaniu polecenia **return**
  lub po wykonaniu sekwencji wszystkich instrukcji zawartych w ciele funkcji,

- jeżeli funkcja jest typu **void**, to używamy samego słowa **return**,
  bez żadnego wyrażenia po nim,

- jeżeli funkcja jest typu innego niż **void** to po poleceniu return musi się pojawić
  wyrażenie odpowiedniego typu (może być w nawiasach), np.:

  return liczba_1; *lub* **return**(liczba_1) ;
  (wyjątek stanowi funkcja główna main)

- nie jest możliwe definiowanie funkcji wewnątrz funkcji

- argumenty funkcji przekazywane są przez wartość, nie przez nazwę,
  z wyjątkiem tablic

ZMIENNE LOKALNE A GLOBALNE

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

int r = 0;                  // zmienna globalna

void wyswietl(int b)
{
    int w = 3;              // zmienna lokalna
    cout << b << endl;
    cout << w << endl;
    cout << r << endl;
 // cout << a;  błąd - zmienna nie jest dostępna

    w += 10;
    r++;
}

int main(void)
{
    int a = 8, b = 12;  // zmienne lokalne
    cout << r;                              // 0
 // cout << w;  błąd - zmienna nie istnieje

    wyswietl(a);                            // 8 3 0

    wyswietl(b);                            // 12 3 1

    wyswietl(17);                           // 17 3 2
}
```