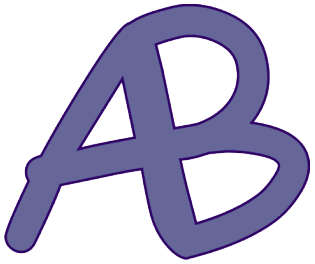
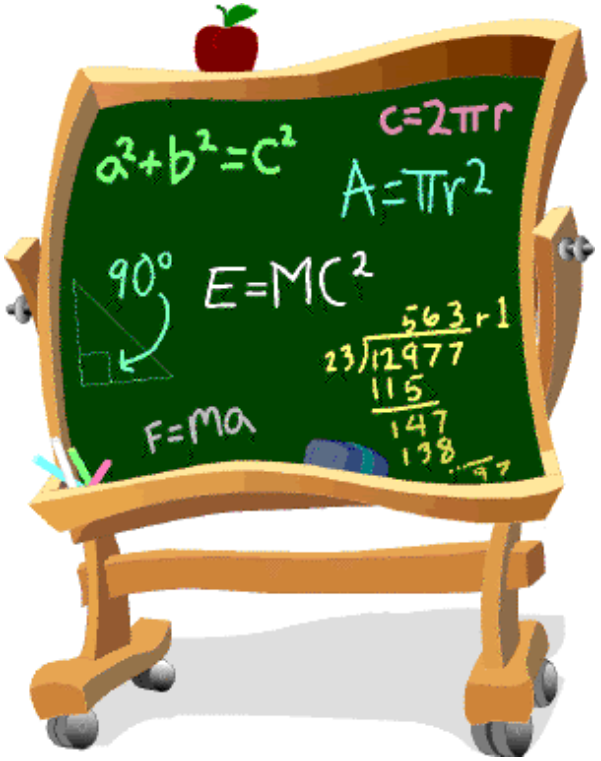


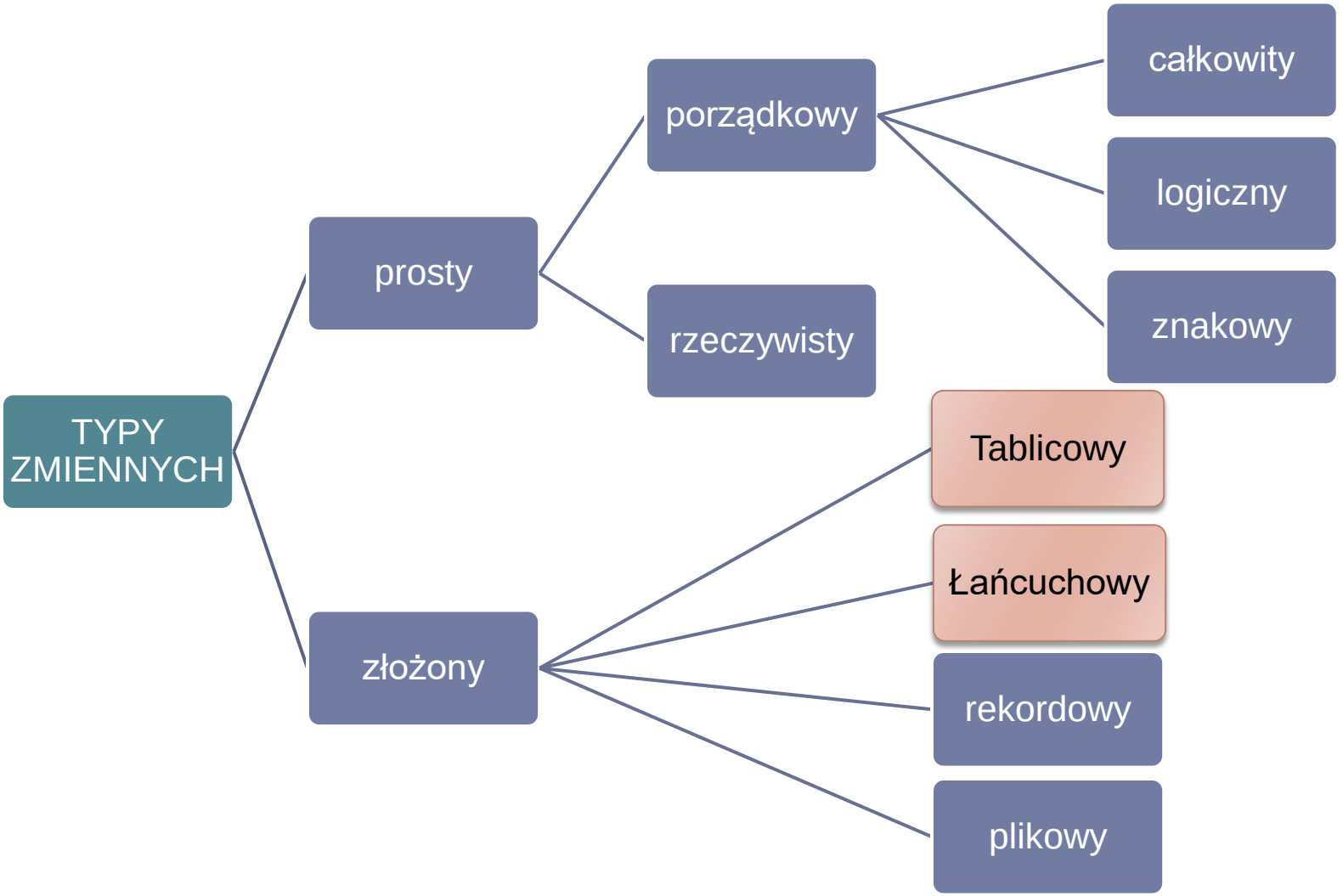
# Wykład: 4

## Tablice statyczne

# Tablice



# Klasyfikacja zmiennych statycznych



# Rodzaje tablic

**Tablice jednowymiarowe (wektory)**

- są zespołem określonej liczby zmiennych o wspólnej nazwie, które ponumerowano liczbami naturalnymi – każda z nich ma przypisany na stałe tzw. indeks,
- mogą przechowywać nie większą od ich długości liczbę elementów zbioru danych jednakowego typu.

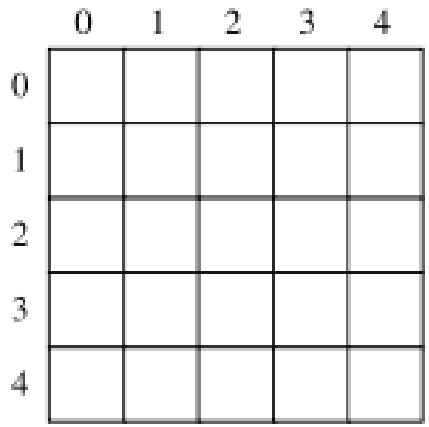
| Tab | | | | | | | | - nazwa tablicy |
|---|---|---|---|---|---|---|---|---|
| 34 | 56 | 32 | -8 | 45 | 2 | 0 | 13 | - wartości |
| Tab(0) | Tab(1) | Tab(2) | Tab(3) | Tab(4) | Tab(5) | Tab(6) | Tab(7) | - pole – nazwa(indeks) |

W zapisie symbolicznym ***T*(6)** oznacza 6 zmienną w tablicy ***T***
Indeks może być określony przez bezpośrednie podanie wartości w odwołaniu do elementu tablicy, np. ***T*(6)**, lub użycie nazwy zmiennej o typie zgodnym z indeksem, np. ***T*(*X*)**. Zmienną ***X*** nazywamy wtedy zmienną indeksową i wskazanie elementu tablicy wymaga odczytania jej aktualnej wartości.

# Rodzaje tablic

**Tablice dwu – i więcej wymiarowe (macierze)**

- są zespołem określonej liczby zmiennych o wspólnej nazwie, które oznaczono dwoma lub więcej indeksami,
- mogą przechowywać nie większą od ich rozmiaru liczbę elementów zbioru danych jednakowego typu.

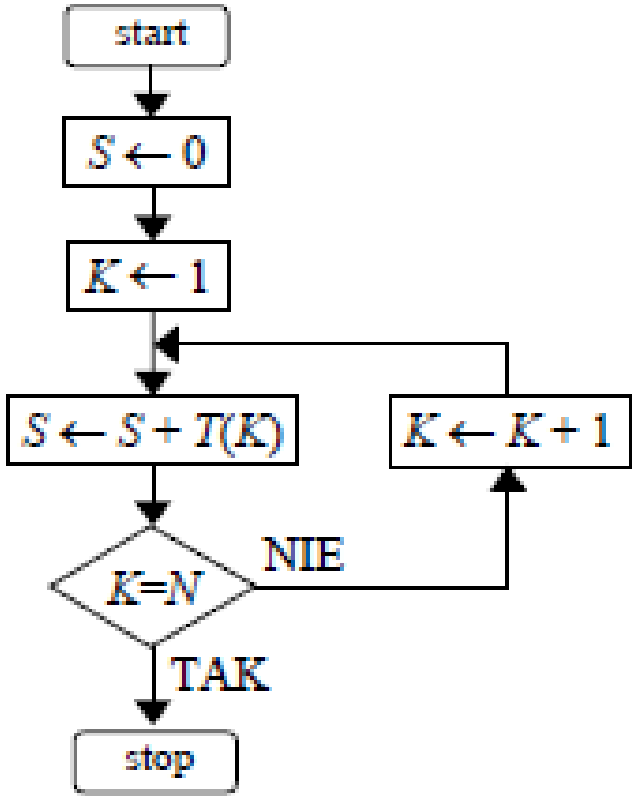



W zapisie symbolicznym ***W*(3, 5)** oznacza zmienną w tablicy ***W*** położoną umownie na przecięciu 3. wiersza i 5. kolumny.

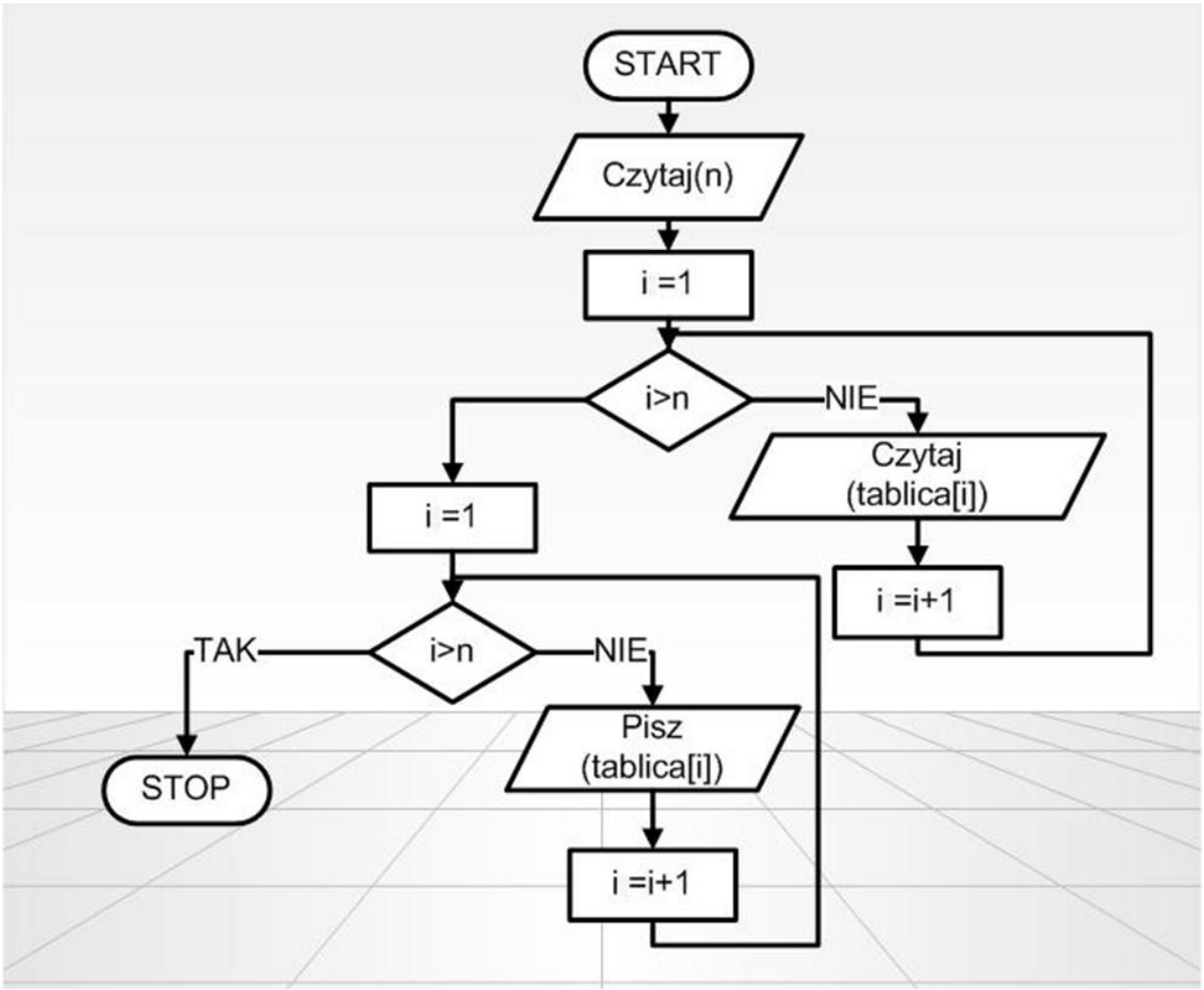
# Obsługa tablicy jednowymiarowej

Algorytm sumowania $N$ liczb zapamiętanych w tablicy $T$

1. $S \leftarrow 0$ (ustalenie początkowej wartości sumy);
2. $K \leftarrow 1$ (ustalenie początkowej wartości zmiennej indeksowej);
3. wykonaj co następuje $N$ razy:
   - 3.1. $S \leftarrow S + T(K)$;
   - 3.2. $K \leftarrow K + 1$.



# Obsługa tablicy jednowymiarowej



# Tablice

Tworzenie jednowymiarowych tablic zmiennych - za deklaracją zmiennej podamy liczbę elementów.

typ_zmiennej nazwa_zmiennej [liczba_elementow];

liczba_elementów - musi być wartością stałą dosłowną, lub stałą const

Np.:

```
int Tablica[ 10 ];
```

Lub:

```
const STALA = 10;
int Tablica[ STALA ];
```

# Tablice

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       int liczbyCalkowite[5];
7           // Definicja tablicy pieciu liczb całkowitych
8       double liczbyRzeczywiste[3];
9           // Definicja tablicy trzech liczb zmiennoprzecinkowych
10      return 0;
11  }
```

# Tablice

Tablice można tworzyć z:

- ✓ typów fundamentalnych (z wyjątkiem void),
- ✓ typów wyliczeniowych (enum),
- ✓ wskaźników,
- ✓ innych tablic;
- ✓ obiektów typu zdefiniowanego przez użytkownika (czyli klasy),
- ✓ wskaźników do pokazywania na składniki klasy.

# Tablice

## Numeracja elementów tablicy zaczyna się od zera.

$$\underbrace{[0][1][2]\cdots[n-1]}$$

*indeksy komórek* n-*elementowej tablicy*

Jeśli zdefiniujemy tablicę:

```
int tab[5];
```

jest to tablica pięciu elementów typu int. Poszczególne elementy tej tablicy to:

```
tab[0]  tab[1]  tab[2]  tab[3]  tab[4]
```

# Inicjalizacja tablic

Inicjalizacja tablicy - nadanie wartości początkowych w momencie definicji tablicy.

Np.:

```
int tab[6] = {2, 3, 5, 7, 11, 13 } ;
```

Jest równoznaczne z:

```
tab[0]=2;  tab[l]=3;   tab[2]=5;
tab[3]=7;  tab[4]=11;  tab[5]=13;
```

Uwaga: zapis: cout << tab[6];
odnosi się do nieistniejącego elementu tablicy.

# Inicjalizacja tablic

Przy zapisie:

```
int tab[] = {2, 3, 5, 7, 11, 13 };
```

kompilator „domyśli się”, że chodzi tablicę 6-cio elementową.

---

Przy zapisie:

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

pierwsze trzy elementy zostaną zainicjalizowane podanymi wartościami, pozostałe zerami:

```
tab[0]=2;  tab[l]=3;  tab[2]=5;
tab[3]=0;  tab[4]=0;  tab[5]=0;
```

# Inicjalizacja tablic

Uwaga: Code::Blocks dopuszcza konstrukcję:

```
typ_elementów nazwa_tablicy[zmienna];
```

pozwala ona na tworzenie statycznych tablic o liczbie elementów podanej w zmiennej. Na przykład:

```
int n;
cin >> n;
double a[n];
```

Nie jest to standardowe rozwiązanie i może nie być przenośne na inne kompilatory C++.

# Obsługa tablicy jednowymiarowej

for (i=0; i < rozmiar; i++) ……;

Przykład:
Dana jest 6-cio elementowa tablica,
Wypisz jeśli wartość w tablicy jest parzysta

```
int tab[6] = {1,2,3,4,5,6};
for(int i=0;i<5;i++)
    if(tab[i]%2==0)
      cout<<tab[i]<<" ";
```

# Inicjalizacja tablic

## Przykład: Lotto

```
1   // Lotto - uŷycie prostej tablicy liczb
2   #include<cstdlib>
3   #include<iostream>
4   #include<ctime>
5   using namespace std;
6
7   const unsigned ILOSC_LICZB = 6;
8   const int MAKSYMALNA_LICZBA = 49;
9
10  int main()
11  {
12      // deklaracja i wyzerowanie tablicy liczb
13      unsigned Liczby[ILOSC_LICZB];
14      for (int i = 0; i < ILOSC_LICZB; ++i)
15          Liczby[i] = 0;
16
```

# Inicjalizacja tablic

## Przykład: Lotto c.d.

```
17  // losowanie liczb
18  //srand (static_cast<int>(time(NULL)));
19  srand(time(NULL));
20  for (int i = 0; i < ILOSC_LICZB; )
21  {
22        // wylosowanie liczby
23        Liczby[i] = rand() % MAKSYMALNA_LICZBA + 1;
24        unsigned main::Liczby
25        // sprawdzenie, czy si« ona nie powtarza
26        bool PowtarzaSie = false;
27        // typ bool istnieje tylko w języku C++
28        // w AnNSI C możemy posłużyć się
29        // typedef enum {TRUE = 1, FALSE = 0} bool;
```

# Inicjalizacja tablic

## Przykład: Lotto c.d.

```
30              for (int j = 0; j < i; ++j)
31              {
32                    if (Liczby[j] == Liczby[i])
33                    {
34                          PowtarzaSie = true;
35                          break;
36                    }
37              }
38
39              // jeżeli si« nie powtarza, przechodzimy do nast«pnej liczby
40              if (!PowtarzaSie) ++i;
41        }
42
43        // wyœwietlamy wylosowane liczby
44        cout << "Wyniki losowania:" << endl;
45        for (int i = 0; i < ILOSC_LICZB; ++i)
46              cout << Liczby[i] << " ";
47  }
```

# Przekazywanie tablic do funkcji

**NAZWA TABLICY**

**jest równocześnie**

**ADRESEM JEJ ZEROWEGO ELEMENTU**

Dla: int tab[10];

zapis: tab jest równoznaczny z &tab[0]

Przekazując tablicę do funkcji w rzeczywistości przekazujemy funkcji wskaźnik do tej tablicy.

# Przekazywanie tablic do funkcji

**Tablicy nie można przesłać przez wartość.**

Można tak przesłać pojedyncze jej elementy, ale nie całość.

Mamy funkcję o nagłówku:

```
void funkcja   (float tab[]);
```

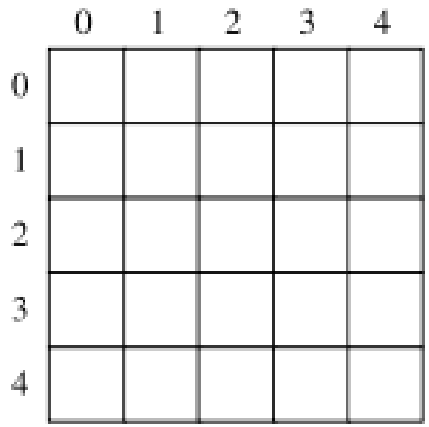
która spodziewa się jako argumentu: tablicy liczb typu float, Taką funkcję wywołujemy na przykład tak:

```
float tablica[4]={ 7, 8.1, 4, 4.12};
funkcja (tablica);
```

# Tablice wielowymiarowe

**Tablice dwu – i więcej wymiarowe (macierze)**

- są zespołem określonej liczby zmiennych o wspólnej nazwie, które oznaczono dwoma lub więcej indeksami,
- mogą przechowywać nie większą od ich rozmiaru liczbę elementów zbioru danych jednakowego typu.




W zapisie symbolicznym ***W*(3, 5)** oznacza zmienną w tablicy ***W*** położoną umownie na przecięciu 3. wiersza i 5. kolumny.

# Tablice wielowymiarowe

W języku C++ tablice wielowymiarowe to tablice, których elementami są inne tablice.

```
int tab_2D[n][m];
```

Definicja ta oznacza: tab_2D jest tablicą n-elementową, z których każdy jest m-elementową tablicą (liczb typu int).

Uwaga: zapis ~~int tab_2D[n , m]~~ jest błędny.

# Tablice wielowymiarowe

Przykład:

```
int tab_2D[4][3];
```

Gdzie: tab_2D jest tablicą 4-elementową, z których każdy jest 3-elementową tablicą liczb typu int.

| [4][3] | 0 | 1 | 2 |
|---|---|---|---|
| 0 | [0] [0] | [0] [1] | [0] [2] |
| 1 | [1] [0] | [1] [1] | [1] [2] |
| 2 | [2] [0] | [2] [1] | [2] [2] |
| 3 | [3] [0] | [3] [1] | [3] [2] |

# Tablice wielowymiarowe

Elementy takie umieszczane są kolejno w pamięci komputera tak, że najszybciej zmienia się najbardziej skrajny prawy indeks.

Stąd, inicjalizacja zbiorcza:

```
int tab[3][2]  = {1,2,3,4,5,6};
```

spowoduje, że elementom tej tablicy zostaną przypisane wartości początkowe tak, jakbyśmy to robili grupą instrukcji:

```
tab[0][0]   = 1;
tab[0][1]   = 2;
tab[1][0]   = 3;
tab[1][1]   = 4;
tab[2][0]   = 5;
tab[2][1]   = 6;
```

# Tablice wielowymiarowe

## Obsługa tablicy dwuwymiarowej:

```
1  #include <iostream>
2
3  using namespace std;
4  int tab[5][3] = { 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
5  int main()
6  {
7
8      for (int j=0; j<5; j++)
9      {
10
11       for (int i=0; i<3; i++)
12        cout << tab[j][i>] << "   ";
13       cout << "\n";
14     }
15     return 0;
16 }
```

```
1   2   3
4   5   6
7   8   9
10   11   12
13   14   15
```

# Literatura:

## W prezentacji wykorzystano przykłady i fragmenty:

- Gręboszj J. : ***Symfonia C++, Programowanie w języku C++ orientowane obiektowo***, Wydawnictwo Edition 2000.
- Jakubczyk K.: *Turbo Pascal i Borland C++ Przykłady*, Helion.

## Warto zajrzeć także do:

- Sokół R. : ***Microsoft Visual Studio 2012 Programowanie w Ci C++,*** Helion.
- Kerninghan B. W., Ritchie D. M.: ***język ANSI C***, Wydawnictwo Naukowo Techniczne.

## Dla bardziej zaawansowanych:

- Gręboszj J. : ***Pasja C++,*** Wydawnictwo Edition 2000.
- Meyers S.: ***język C++ bardziej efektywnie***, Wydawnictwo Naukowo Techniczne