Tablice

1

Tablice jednowymiarowe

Ciąg elementów tego samego typu.

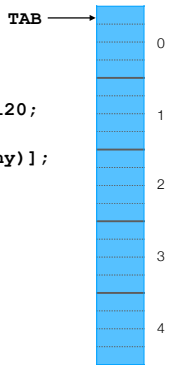
```
typ  identyfikator [ rozmiar ] ;
```

typ : liczbowy, znakowy, wskaźnikowy lub typ strukturalny

rozmiar: wyrażenie stałe o wartości większej od zera
(w gcc wyrażenie całkowitoliczbowe)

2

```
int TAB[5]; // 0, 1, 2, 3, 4

const int sumy = 50, iloczyny = 120;

float WYNIKI[ 2 * (sumy + iloczyny)];

int i;

i = TAB[3];

// TAB + 3 * sizeof int
```



3

```
WYNIKI[i + 2] = WYNIKI[i] + WYNIKI[i + 1];
```

```
double TAB_DANYCH [ 7 ] ;

for (int ix = 0; ix < 12; ++ix ){
    TAB_DANYCH [ ix ] = 48.74;
}

// ?
```

4

Wartości początkowe

5

```
typ identyfikator [rozmiar] = {lista_ wartości};
```

lista_wartości : wyrażenia stałe

```
long MM[3] = { 154835L, 337782L, 0L };

/*
   MM[0] == 154835,
   MM[1] == 33782,
   MM[2] == 0
*/
```

6

```
const float F1 = 3.5E7F, F2 = 33.E8F;

float DANE[ ] = { F1 + F2, F1 / (F1 - F2)};

/* tablica DANE ma 2 elementy */


double PQ[150] = { 1.5, 3.8, 3.8, 2.7 };

/* PQ[0] == 1.5, PQ[1] == 3.8, PQ[2] == 3.8,
   PQ[3] == 2.7, pozostałe  elementy == 0 */


double Empty[1200] = {0};    // zerowanie
```

7

```
char NN[5] = { "alfa" };  // NN[4] == 0

char MM[4] = { "beta" };  // błąd, tablica zbyt mała


int i = 1, j = 2, k;

int *WW[3] = { &i, &j };

WW[2] = &k;

*WW[2] = *WW[0] + *WW[1];     // k == 3


char *TYDZIEN[ ] = { "pon", "wto", "sro",
                     "czw", "pia", "sob", "nie" };
```

Wskaźniki

8

Zapisy **`*TAB`** i **`TAB [ 0 ]`** są równoważne.

```
char bufor[8];
char *pp;

pp = bufor;
// jest równoważne
pp = & bufor[0];
```

Zapisy `*( TAB + 4 )` i `TAB [ 4 ]` są równoważne

```
const int ele = 25;
short     TS[ele];
int       TI[ele];
double    TD[ele];

for (int i = 0; i < ele; ++i)
{
    *(TS + i) = 1;
    *(TI + i) = 1;
    *(TD + i) = 1.0;
}
```

9

Wczytywanie elementów tablicy jednowymiarowej

```
int TabLicz [ 125 ];

for (int i = 0; i < 125; ++i){
    scanf("%d", &TabLicz[ i ]);
}
```

```
double Rzeczywiste [ 12 ];

for (int i = 0; i < 12; ++i){
    scanf("%lf", &Rzeczywiste[ i ]);
}
```

10

Rozmiar

```
long A[5];
int obszar, element, elementow;

obszar    = sizeof A;               // == 20

element   = sizeof A[0];            // == 4

elementow = sizeof A / sizeof A[0]; // == 5
```

11

```
int dl;

scanf("%d", &dl);

double Tab[ dl ];// błąd, nie można inicjować
```

```
int T[5] = { 3, [2] = 5, [4] = 2 };

for (int i = 0; i < 5; ++i){
    printf("\t%d", T[i]);
}

//  3  0  5  0  2
```

12

Tablice wielowymiarowe

```
float MACIERZ[10][20];/* 10 wierszy, 20 kolumn
                         dostęp MACIERZ[nw][nk] */


const int kwa = 30;
long KWADRAT[kwa][kwa];


for (int i = 0; i < kwa; ++i)
    for (int j = 0; j < kwa; ++j)
        KWADRAT [ i ] [ j ] = i == j ? 0 : 1;

/* główna przekątna zostanie wyzerowana,
   pozostałe elementy == 1 */
```

13

```
int BB[2][3] = { { 1, 2, 3} , {4, 5, 6} };
/*    BB[0][0] == 1, BB[0][1] == 2, BB[0][2] == 3
      BB[1][0] == 4, BB[1][1] == 5, BB[1][2] == 6 */


float CC[3][2] = { { 8.5 } , { 3.2 } };
/*    CC[0][0] == 8.5, CC[0][1] == 0
      CC[1][0] == 3.2, CC[1][1] == 0
      CC[2][0] == 0  , CC[2][1] == 0   */


long   LL[2][3];
sizeof LL;                    // 24
sizeof LL[0];                 // 12
sizeof LL[0][0];              // 4
```

14

Wczytywanie macierzy wierszami

```
const int Wie = 10, Kol = 5;
int MM [ Wie ][ Kol ];

for (int i = 0; i < Wie; ++i)
   for (int j = 0; j < Kol; ++j)
      scanf("%d", & MM[ i ][ j ];


int ile = Wie * Kol, k = 0;
int *p = &MM[0][0];

while (k++ < ile)   // liniowa pamięć
   scanf("%d", p++);
```

15

```
float TTT[5][10][20];

/* tablica trójwymiarowa składająca się
z 5-ciu macierzy o 10 wierszach
i 20 kolumnach każda */

for (int mac = 0; mac <  5; mac ++)
    for (int wie = 0; wie < 10; wie ++)
        for (int kol = 0; kol < 20; kol ++)
            TTT[mac][wie][kol] = 1.0F;
```

16