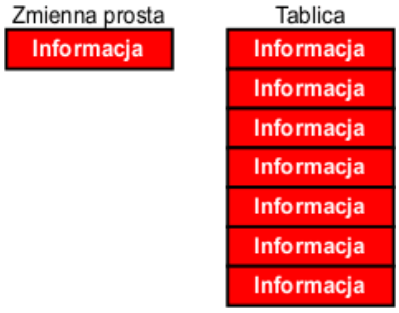
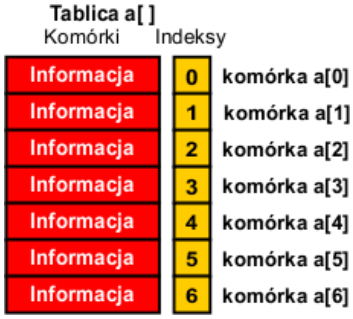
Tablice

Tablica (ang. array) jest złożoną strukturą danych, która składa się z ciągu elementów tego samego typu. Są one często wykorzystywane ze względu na ich użyteczność. W zmiennej prostej możemy umieścić tylko pojedynczą informację, np. liczbę lub literę. W tablicy takich informacji można umieścić więcej. Ważne jest zatem ich poznanie oraz zrozumienie sposobu dostępu do danych umieszczonych w tablicy.



Tablica zbudowana jest z komórek, które możemy traktować jak zmienne proste (chociaż - jak zawarto w dalszej części - komórka tablicy może być nową tablicą). Komórki te są ponumerowane kolejnymi liczbami, które nazywamy **indeksami**. Numeracja indeksów w języku C rozpoczyna się od wartości 0, a indeks jest liczbą całkowitą nieujemną. Indeksy umieszczamy w klamrach kwadratowych obok nazwy tablicy. Wszystkie komórki są tego samego typu.



Dzięki tym indeksom możemy się odwoływać do wybranej komórki tablicy. Jeśli zmienna *a* jest tablicą i zawiera 10 komórek, to nazwy tych komórek są następujące:

a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]

Zastosowanie tablic

Tablice wykorzystywane są najczęściej gdy mamy do czynienia z serią danych, ciągiem, np.:

- Tworzymy program, który pobierze od użytkownika 100 liczb i je zapamięta. Następnie na podstawie tych liczb wyznaczy różne statystyki – minimum, maksimum, średnią, medianę, itp.
- Tworzymy aplikację umożliwiającą sortowanie danych (liczbowych lub tekstów). Wówczas również potrzebujemy struktury, która zapamięta dane podlegające sortowaniu.

Tablice jednowymiarowe

Tablica jednowymiarowa (inaczej wektor) ma postać jak na powyższym rysunku. Podobnie jak zmienna prosta - reprezentowana jest przez nazwę (identyfikator) ale w jej wnętrzu znajdować się może wiele komórek.

Tworzenie tablic jednowymiarowych

Tablice jednowymiarowe możemy tworzyć na 2 sposoby:

- Deklaracja tablicy z podaniem typu przechowywanych danych oraz rozmiaru struktury:

```c
#include<stdio.h>
int main()
{
    int iTab[5]; // 5-celementowa tablica liczb calkowitych
    float fTab[10]; // 10-elementowa tablica liczb zmiennoprzecinkowych
    char chTab[3]; // 3-elementowa tablica znakow
    return 0;
}
```

- Deklaracja tablicy z podaniem typu przechowywanych danych oraz wypełnieniem jej elementami – bez konieczności podawania rozmiaru:

```c
#include<stdio.h>
int main()
{
    int iTab2[] = {1,3,-4,0,6}; // 5-elementowa tablica liczb calkowitych
    float fTab2[] = {2.3,5.1,-4.3}; // 3-elementowa tablica liczb zmiennoprzecinkowych
    char chTab2[] = {'q','b','a','t'}; // 4-elementowa tablica znakow
    return 0;
}
```

W wyniku wykonania linii kodu: `int iTab[5];` zostanie utworzona tablica o nazwie *iTab*, która będzie zawierała 5 komórek o indeksach: iTab[0], iTab [1], iTab [2], iTab [3], iTab [4]. W każdej komórce tablicy będzie można umieścić liczbę całkowitą ze znakiem.

UWAGA: Rozmiar tablicy *iTab* wynosi 5 – tyle mamy w niej komórek. Natomiast największy indeks tej tablicy to 4 – indeksujemy komórki licząc od 0. Zatem liczba komórek tablicy jest o 1 większa od numeru ostatniego indeksu tablicy.

W przypadku wykonania linii kodu: `int iTab2[] = {1,3,-4,0,6};` również utworzona zostanie tablica o rozmiarze 5-ciu komórek. Dodatkowo każda z komórek zostanie wypełniona wartością typu *int*. Mamy zatem deklarację tablicy połączoną z jej inicjalizacją.

Wstawianie elementów do tablicy oraz pobieranie elementów z tablicy

W przypadku wykonania instrukcji: `int iTab[5];` powstała tablica int-ów o rozmiarze 5, ale nie umieściliśmy w niej żadnych wartości. Możemy tego dokonać odwołując się do każdego indeksu tablicy i nadając mu wartość:

```c
#include<stdio.h>
int main()
{
    int iTab[5];
    iTab[0] = 5;
    iTab[1] = 3;
    iTab[2] = -5;
    iTab[3] = 6;
    iTab[4] = 0;
    return 0;
}
```

Aby pobrać element (wartość) tablicy przypisaną do konkretnej komórki należy odwołać się do nazwy tablicy oraz podać indeks komórki, której wartość chcemy uzyskać. Pozyskaną wartość można przypisać do zmiennej od razu użyć np. wypisując ją na ekran.

```c
#include<stdio.h>
int main()
{
    int iTab[] = {5,3,-5,6,0};

    int element0 = iTab[0]; // pozyskanie wartosci elementu i przypisanie do zmiennej
    printf("iTab[0]= %d\n", element0); // wypisanie wartosci zmiennej

    printf("iTab[0]= %d\n", iTab[0]); // wypisanie wartosci elementu tablicy

    return 0;
}
```

Aby wypisać całą zawartość tablicy można posłużyć się pętlą:

```c
#include<stdio.h>
int main()
{
    int iTab[] = {5,3,-5,6,0}; // tablica 5-elementowa
    int i; // zmienna reprezentujaca indeks tablicy
    for(i=0; i<=4; i++) // zaczynamy od indeksu 0 i konczymy na 4
        printf("iTab[%d]= %d\n", i, iTab[i]);
    return 0;
}
```

Dobrą praktyką przy korzystaniu z tablic może być utworzenie stałej reprezentującej rozmiar tablicy, aby np. w sytuacji konieczności wypisania jej elementów w łatwy sposób móc określić warunki pętli:

```c
#include<stdio.h>
#define N 5
int main()
{
    int iTab[N];
    iTab[0] = 5;
    iTab[1] = 3;
    iTab[2] = -5;
    iTab[3] = 6;
    iTab[4] = 0;
    int i;
    for(i=0; i<N; i++)
        printf("iTab[%d]= %d\n", i, iTab[i]);
    return 0;
}
```

Rozmiar tablicy możemy również obliczyć. W tym celu należy pobrać rozmiar tablicy zajmowany w pamięci komputera i podzielić przez rozmiar jaki zajmuje w pamięci komputera typ danych umieszczonych w tablicy:

```c
#include<stdio.h>
int main()
{
    int iTab[] = {5,3,-5,6,0};
    int rozmiar = sizeof(iTab)/sizeof(int); // wyznaczenie rozmiaru tablicy
    printf("rozmiar tablicy iTab[]= %d\n", rozmiar);
    int i;
    for(i=0; i<=rozmiar-1; i++)
        printf("iTab[%d]= %d\n", i, iTab[i]);
    return 0;
}
```

ZADANIE

Przetestuj działanie powyższych przykładów dotyczących wstawiania elementów i pobierania ich wartości. Zwróć szczególną uwagę na konstrukcję warunków użytych w pętlach użytych do wypisania zawartości tablic.

Tablice znaków

Bardzo użyteczne są tablice typu *char*. Komórki tych tablic zawierają znaki tekstu. Tekst w języku C jest tworzony ze znaków o kodach ASCII, które tworzą ciąg liter. Przyjęto konwencję, że za ostatnim znakiem tekstu umieszczany jest kod o wartości 0. Nosi on nazwę EOT (ang. End Of Text – Koniec Tekstu). Poniższa definicja:

```
char t[] = "Jasiek";
```

tworzy tablicę tekstową o nazwie *t*, która w poszczególnych komórkach zawiera kolejne literki tekstu:

komórki	t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]
zawartość	J	a	s	i	e	k	EOT

Należy zauważyć, że tablica zdefiniowana za pomocą tekstu zawsze ma rozmiar o 1 większy od liczby znaków w tekście, aby pomieścić EOT.

Poniższy przykład obrazuje w jaki sposób tekst „Jasiek” jest rozumiany przez program jako tablica znaków. W przykładzie wypisywany jest rozmiar tablicy oraz każdy ze znaków wraz z jego kodem w tablicy ASCII:

```
#include<stdio.h>
int main()
{
    char t[] = "Jasiek";
    int rozmiar = sizeof(t)/sizeof(char);
    printf("rozmiar tablicy t[]= %d\n", rozmiar);
    int i;
    for(i=0; i<=rozmiar-1; i++)
        printf("iTab[%d]= %c\tkod ASCII: %d\n", i, t[i], t[i]);
    return 0;
}
```

```
rozmiar tablicy t[]= 7
iTab[0]= J      kod ASCII: 74
iTab[1]= a      kod ASCII: 97
iTab[2]= s      kod ASCII: 115
iTab[3]= i      kod ASCII: 105
iTab[4]= e      kod ASCII: 101
iTab[5]= k      kod ASCII: 107
iTab[6]=        kod ASCII: 0
```

Wynik działania programu rzeczywiście potwierdza, że ostatnim znakiem w tablicy *t* jest EOT, którego kod ASCII wynosi 0.

Wiemy już, że tekst jest tablicą znaków, a tablica znaków może być rozumiana jako tekst. Zatem chcąc wypisać elementy tablicy typu *char* możemy posłużyć się znakiem przekształcenia *%s*:

```
#include<stdio.h>
int main()
{
    char t[] = "Jasiek";
    int rozmiar = sizeof(t)/sizeof(char);
    printf("rozmiar tablicy t[]= %d\n", rozmiar);
    int i;
    for(i=0; i<=rozmiar-1; i++)
        printf("iTab[%d]= %c\tkod ASCII: %d\n", i, t[i], t[i]);
    printf("[%s]\n", t);
    return 0;
}
```

ZADANIE

Przetestuj działanie powyższych przykładów dotyczących tablic znaków.

Tablice wielowymiarowe

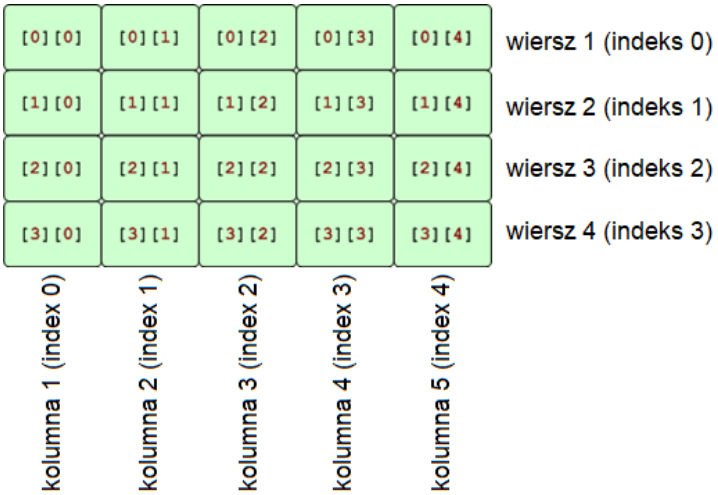
W języku C istnieje możliwość korzystania z tablic, które mają więcej niż jeden wymiar. Przykładem mogą być tablice dwuwymiarowe, które posiadają zarówno wiersze jak i kolumny. Innym przykładem

są tablice trójwymiarowe posiadające wiersze, kolumny i rzędy. Możliwe jest także używanie tablic o większej liczbie wymiarów.

Najczęściej spotykanymi tablicami wielowymiarowymi są tablice dwuwymiarowa, zwane również macierzami lub tablicami prostokątnymi. To właśnie na ich przykładzie objaśnione zostaną zagadnienia związane z tablicami wielowymiarowymi.

Postać tablicy dwuwymiarowej

Tablica dwuwymiarowa – jak wspomniano – posiada zarówno wiersze jak i kolumny i można ją przedstawić w następujący sposób:



Tworzenie tablicy dwuwymiarowej

Tablice dwuwymiarowe – podobnie jak 1-wymiarowe - możemy tworzyć na 2 sposoby:

- Deklaracja tablicy z podaniem typu przechowywanych danych oraz rozmiaru struktury:

```c
#include<stdio.h>
int main()
{
    int iTab[5][4]; // tablica 2-wymiarowa intow 5w x 4k (20 komorek)
    float fTab[10][10]; // tablica liczb zmiennoprzecinkowych 10x10
    char chTab[3][6]; // tablica znakow 3x6
    return 0;
}
```

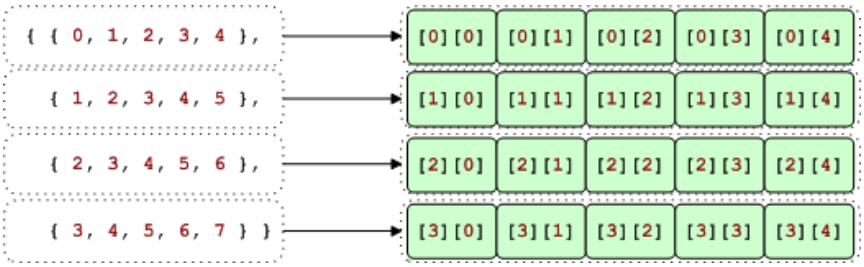
- Deklaracja tablicy z podaniem typu przechowywanych danych, rozmiaru struktury oraz wypełnieniem jej elementami:

```c
#include<stdio.h>
int main()
{
                                // tablica 2-wymiarowa intow 5w x 4k (20 komorek)
    int iTab2[5][4] = {{1,3,-4,0},{1,-3,4,0},{0,3,-4,2},{2,-2,-4,0},{9,-1,-4,0}};
                                // tablica liczb zmiennoprzecinkowych 3x2
    float fTab2[3][2] = {{1.3,2.1},{2.3,4.1},{2.4,5.5}};
                                // tablica znakow 2x4
    char chTab2[2][4] = {{'q','b','a','t'},{'q','b','a','t'}};
    return 0;
}
```

W przypadku pierwszego sposobu deklaracji tablic dwuwymiarowych musimy określić zarówno liczbę wierszy jak i liczbę kolumn dla naszej tablicy. Czynimy to w podwójnych nawiasach kwadratowych umieszczonych po nazwie tablicy.

W drugim sposobie deklarowania tablic dwuwymiarowych uzupełniamy ją od razu wartościami (inicjalizujemy). Należy zauważyć, że również tutaj musimy podać w nawiasach kwadratowych rozmiar

tworzonej tablicy (dla tablic 1-wymiarowych nawiasy pozostawały puste). Wartości, którymi inicjalizujemy tablicę (prawa strona instrukcji przypisania) zostaną rozmieszczone w następujący sposób:



Wstawianie elementów oraz pobieranie elementów z tablicy dwuwymiarowej

Wstawianie elementów do tablicy dwuwymiarowej odbywa się wg następującej zasady:

```c
#include<stdio.h>
int main()
{
    int iTab[3][2];
    iTab[0][0] = 0; // w0 k0
    iTab[0][1] = 1; // wo k1
    iTab[1][0] = 2; // w1 k0
    iTab[1][1] = 3; // w1 k1
    iTab[2][0] = 4; // w2 k0
    iTab[2][1] = 5; // w2 k1
    return 0;
}
```

Aby pobrać element (wartość) tablicy przypisaną do konkretnej komórki należy odwołać się do nazwy tablicy oraz podać indeksy komórki, której wartość chcemy uzyskać. Pozyskaną wartość można przypisać do zmiennej od razu użyć np. wypisując ją na ekran.

```c
#include<stdio.h>
int main()
{
    int iTab[5][4] = {{1,3,-4,0},{1,-3,4,0},{0,3,-4,2},{2,-2,-4,0},{9,-1,-4,0}};

    int element0_0 = iTab[0][0]; // pozyskanie wartosci elementu i przypisanie do zmiennej
    printf("iTab[0]= %d\n", element0_0); // wypisanie wartosci zmiennej

    printf("iTab[0]= %d\n", iTab[0][0]); // wypisanie wartosci elementu tablicy

    return 0;
}
```

Aby wypisać całą zawartość tablicy można posłużyć się podwójną pętlą w celu ‘przechodzenia’ przez poszczególne wiersze i kolumny:

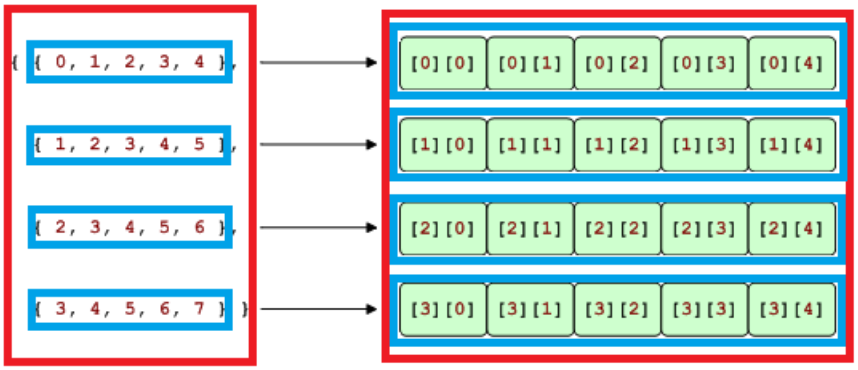
```c
#include<stdio.h>
int main()
{
    int iTab[5][4] = {{1,3,-4,0},{1,-3,4,0},{0,3,-4,2},{2,-2,-4,0},{9,-1,-4,0}};
    int w, k; // zmienne do reprezentacji wiersza i kolumny tablicy
    for(w=0; w<5; w++)
    {
        for(k=0; k<4; k++)
            printf("iTab[%d][%d]= %d\t", w, k, iTab[w][k]); // wypisanie wartosci elementu tablicy
        printf("\n");
    }
    return 0;
}
```

Rozmiar tablicy możemy również obliczyć. W tym celu należy pobrać rozmiar całej tablicy zajmowany w pamięci komputera i podzielić przez rozmiar jaki zajmuje w pamięci komputera typ danych umieszczonych w tablicy. W wyniku dostaniemy liczbę komórek naszej tablicy.

Możliwe jest także obliczenie liczby wierszy i kolumn tablicy dwuwymiarowej. Aby wyznaczyć liczbę kolumn wystarczy obliczyć długość dowolnego wiersza, tj. liczbę komórek (kolumn) w wierszu. W poniższym przykładzie liczba kolumn ustalana jest na podstawie długości wiersza o indeksie 0. Mając rozmiar całej tablicy oraz liczbę jej kolumn w łatwy sposób możemy wyznaczyć liczbę wierszy naszej tablicy.

```c
#include<stdio.h>
int main()
{
    int iTab[5][4] = {{1,3,-4,0},{1,-3,4,0},{0,3,-4,2},{2,-2,-4,0},{9,-1,-4,0}};
    int rozmiar = sizeof(iTab)/sizeof(int); // wyznaczenie rozmiaru calej tablicy
    printf("rozmiar tablicy iTab[][]= %d\n", rozmiar);
    int lK = sizeof(iTab[0])/sizeof(int); // wyznaczenie liczby kolumn tablicy
    printf("l. kolumn tablicy iTab[][]= %d\n", lK);
    int lW = rozmiar/lK; // wyznaczenie liczby wierszy tablicy
    printf("l. wierszy tablicy iTab[][]= %d\n", lW);
    return 0;
}
```

Biorąc pod uwagę fakt możliwości obliczenia liczby kolumn na podstawie długości wiersza możemy stwierdzić, że tablica dwuwymiarowa może być rozumiana jako tablica jednowymiarowa tablic jednowymiarowych (wektor wektorów). Wyraża to poniższa grafika:



Zadania

1. Napisz program, który pobierze od użytkownika 10 liczb i je zapamięta w tablicy, a następnie wyznaczy następujące statystyki dla podanych liczb: suma, minimum, maksimum, średnia, odchylenie standardowe, mediana.
2. Napisz program, który utworzy macierz reprezentującą klasyczną tabliczkę mnożenia 10x10. Wartości poszczególnych komórek mają być wyznaczane i uzupełniane automatycznie przez program. Wyświetl zbudowaną macierz.
3. Napisz program, który realizował będzie mnożenie macierzy przez macierz. Dane wejściowe (macierze) mają być podawane przez użytkownika.
   Zasada działania:

- użytkownik podaje rozmiar tablicy 1 i tablicy 2;
- program sprawdza czy dla takich tablic możliwe jest ich przemnażanie (jeśli tak - tworzy tablicę wynikową o odpowiednim rozmiarze, jeśli nie – wypisuje stosowny komunikat i kończy działanie);
- następnie program prosi użytkownika o podawanie wartości poszczególnych komórek obydwu tablic;
- dalej program uzupełnia tablicę wynikową i wyświetla ją na ekran;

4. Napisz program, który zliczał będzie wystąpienia poszczególnych znaków w zdaniu podanym przez użytkownika. Dla uproszczenia przyjmij, że zliczmy wyłącznie małe litery od a do z oraz spacje, kropki i przecinki.