INFORMATYKA

C

WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI
ELEKTROTECHNIKA
DR INŻ. MICHAŁ ŁANCZONT

VII

Zakres wykładu

1. *Tablica*
2. *Tablica dynamiczna*
3. *Dynamiczna alokacja zmiennych*
4. *Tablica wielowymiarowa*

Tablica

- Klasyczne zmienne deklarowane w języku C (`int`, `float`, `char`) pozwalają na przechowywanie w pamięci pojedynczej wartości danego typu
- Przy konieczności przechowywania większej liczby wartości konieczne by było deklarowanie dla każdej z nich osobnej zmiennej
- Wygodniejszym rozwiązaniem jest przechowywanie zestawu danych jako jednej zwartej grupy powiązanej z jedną nazwą

Tablica

- W języku C Rozwiązanie tak postawionego problemu jest tablica
- Tablica jest zmienna umożliwiającą przechowywanie w pamięci określonej liczby wartości danego typu
- Każdy z elementów tablicy jest indeksowany
- Odwołać się do niego można poprzez podanie nazwy zmiennej tablicowej i numeru indeksu pod jakim dana wartość został w tablicy zapisana
- Tablica `n` elementowa indeksuje wartości od `0` do `n-1`

Deklaracja

- Tablica w języku C jest deklarowana analogicznie jak klasyczna zmienna

```
typ nazwa[rozmiar];
```

- Rozmiar definiuje maksymalną liczbę elementów danej tablicy

- Tablica może przechowywać wartości tylko jednego określonego typu

```
int liczba[20];
float srednia[30];
```

Tablica

- Elementy tablicy indeksowane są liczbami całkowitymi (`int`) z zakresu `0` do `n-1`
- Do poszczególnych elementów tablicy odwołać się można:
- Podając wartość liczbową (`int`)

```
x = liczba[3];
```

- Za pośrednictwem zmiennej całkowitej

```
x = liczba[y];
```

- Poprzez wynik obliczeń (`int`)

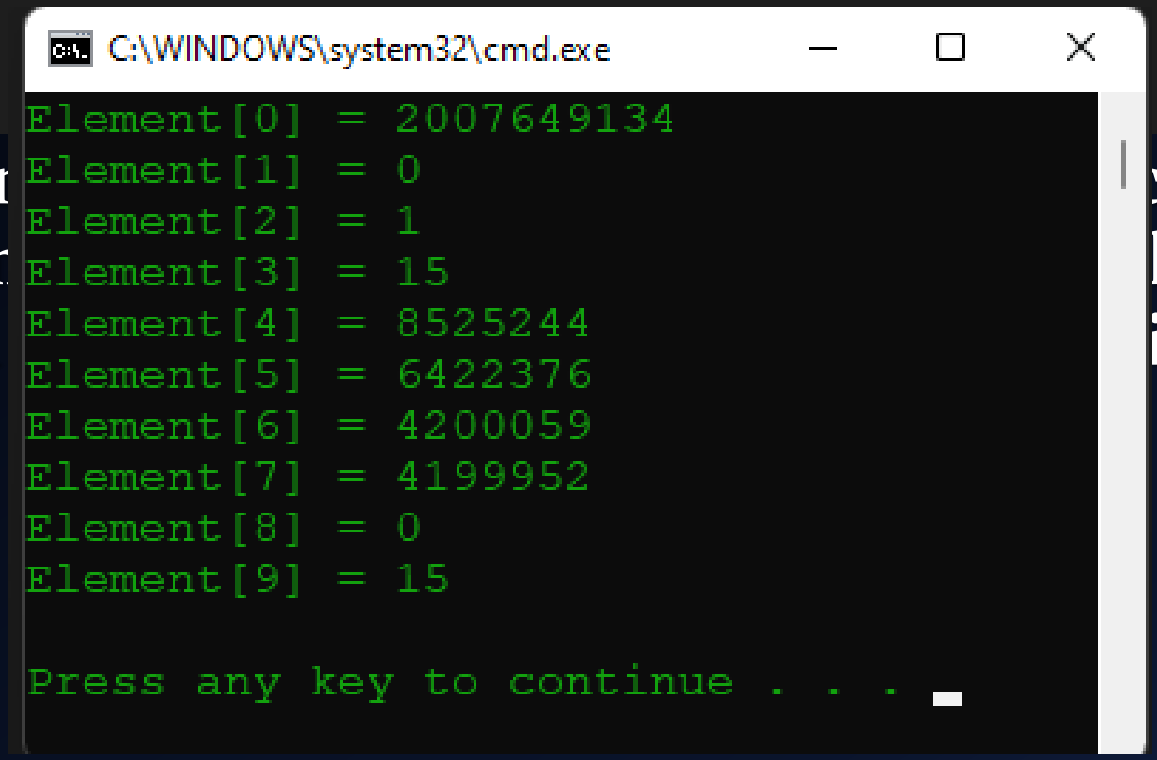
```
x = liczba[n+2];
```

- Poprzez wynik zwracany przez funkcję (`int`)

```
x = liczba[cel()];
```

Tablica

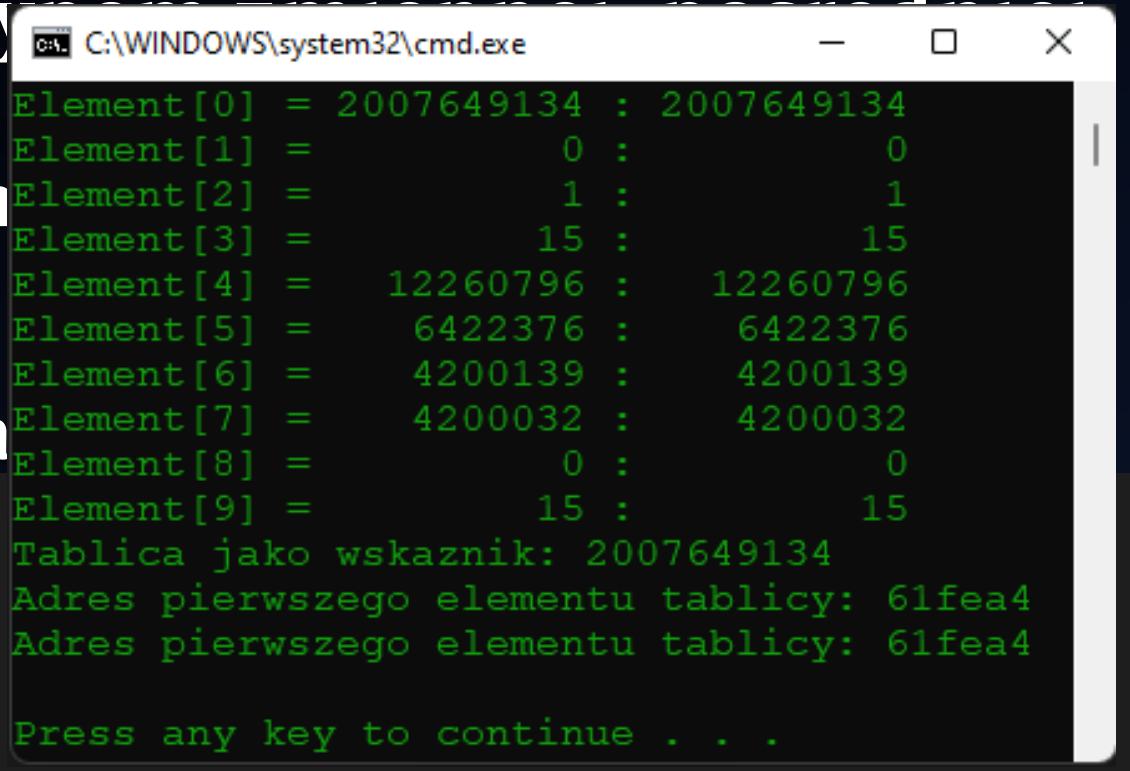
```
1  #include <stdio.h>
2
3  int main()
4  {
5      int liczba[10];
6      for(int i=0;i<10;i++)
7          printf("Element[%i] = %i\n",i,liczba[i]);
8  }
```



- Podobn… …cznych zmienn… …laracji tablica …i

Tablica

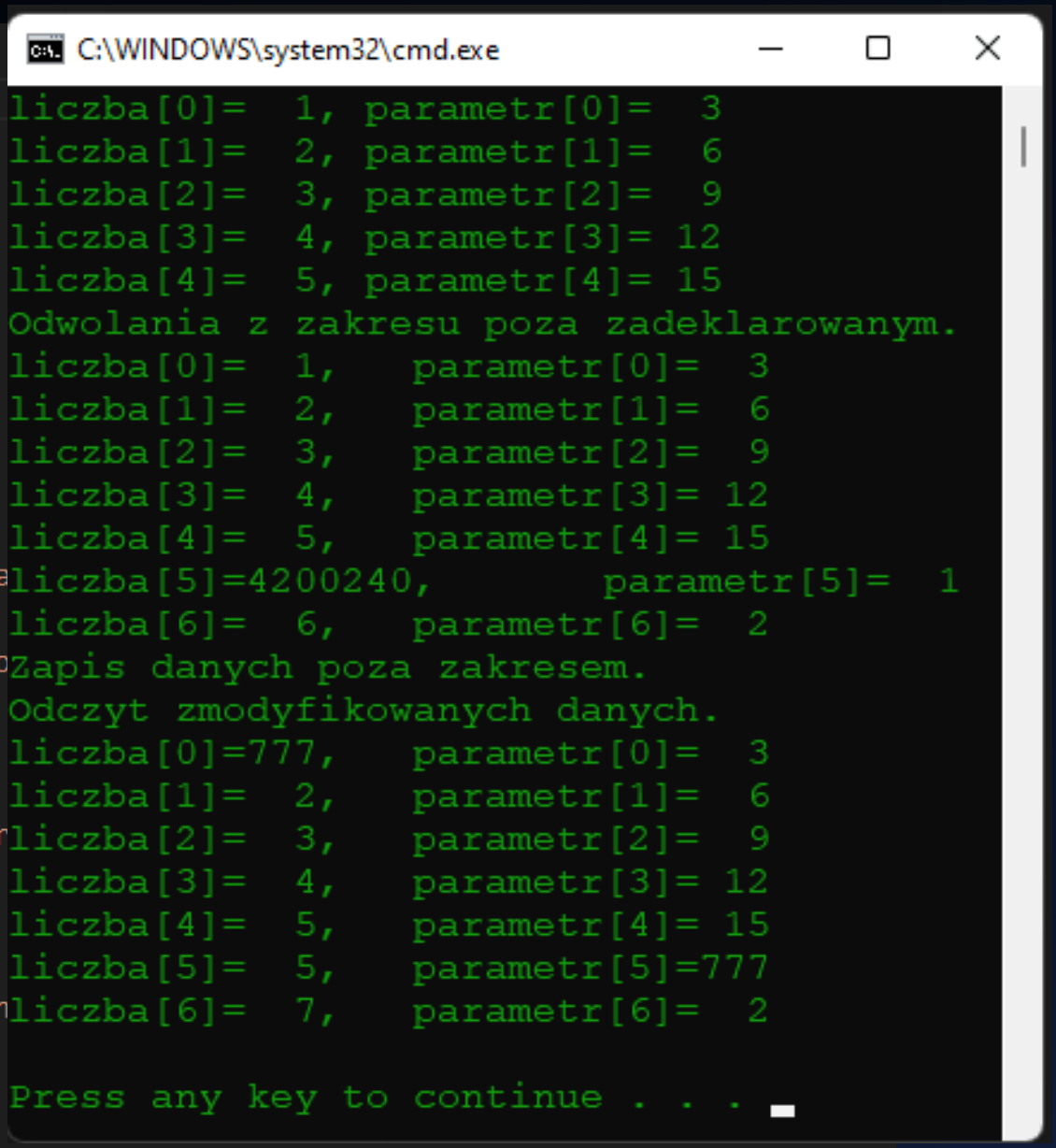
- Tablica jest ty
  pomiędzy
  wskaźnikiem
- Pod nazwą ta




```c
#include <stdio.h>

int main()
{
    int liczba[10];
    for(int i=0;i<10;i++)
        printf("Element[%i] = %10i : %10i\n",i,*(liczba+i),liczba[i]);
    printf("Tablica jako wskaznik: %i\n", *liczba);
    printf("Adres pierwszego elementu tablicy: %x\n",&(liczba[0]));
    printf("Adres pierwszego elementu tablicy: %x\n",liczba);
}
```

Tablica

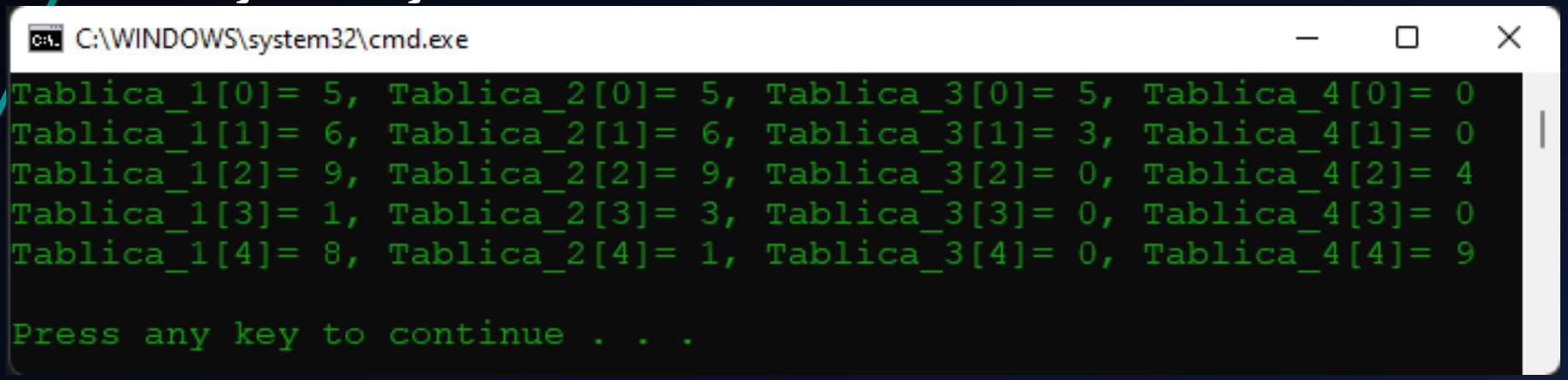
```c
#include <stdio.h>

int main()
{
    int liczba[5], parametr[5];
    for(int i=0;i<5;i++)
    {
        liczba[i] = i+1;
        parametr[i] = 3*(i+1);
    }

    for(int i=0;i<5;i++)
        printf("liczba[%i]=%3i, pa

    printf("Odwolania z zakresu po
    for(int i=0;i<7;i++)
        printf("liczba[%i]=%3i,\t

    printf("Zapis danych poza zakr
    liczba[5]=555;
    parametr[5]=777;

    printf("Odczyt zmodyfikowanych
    for(int i=0;i<7;i++)
        printf("liczba[%i]=%3i,\t
}
```



Tablica - inicjacja

```c
#include <stdio.h>

int main()
{
    int liczby1[5] = {5,6,9,1,8};
    int liczby2[]  = {5,6,9,3,1};
    int liczby3[5] = {5,3};
    int liczby4[]  = {[2]=4,[4]=9};
    for(int i=0;i<5;i++)
        printf("Tablica_1[%i]=%2i, Tablica_2[%i]=%2i, Tablica_3[%i]=%2i, Tablica_4[%i]=%2i\n"
        ,i,liczby1[i],i,liczby2[i],i,liczby3[i],i,liczby4[i]);
}
```

wybranych elementów

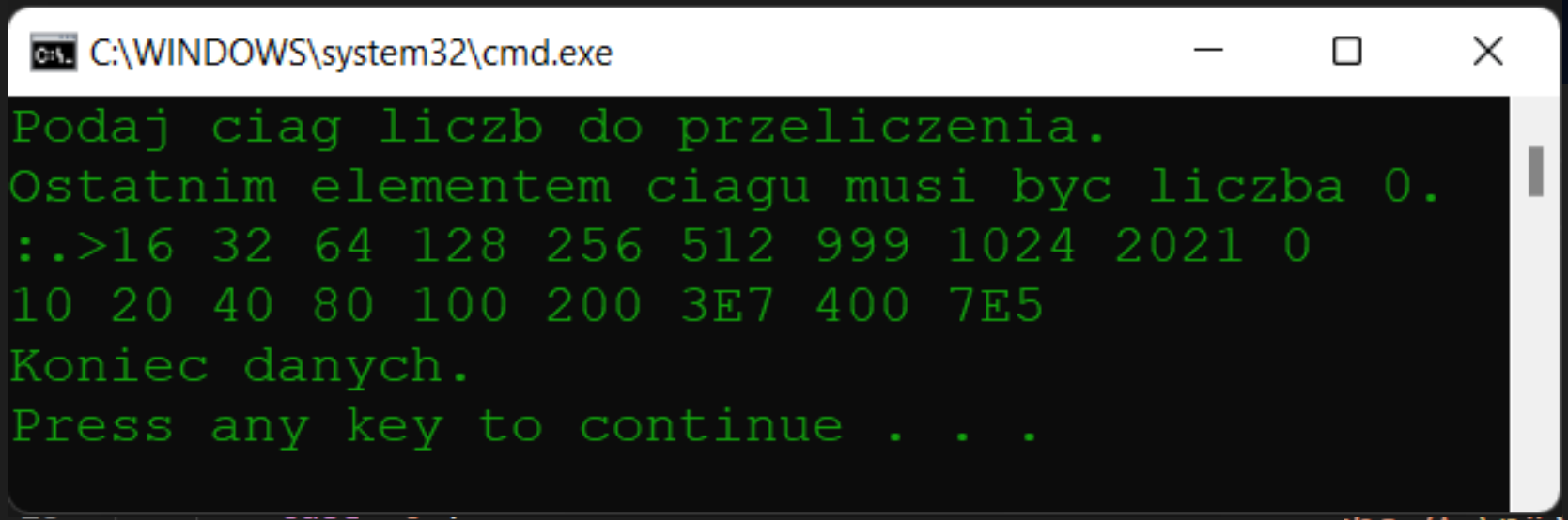


Przykład

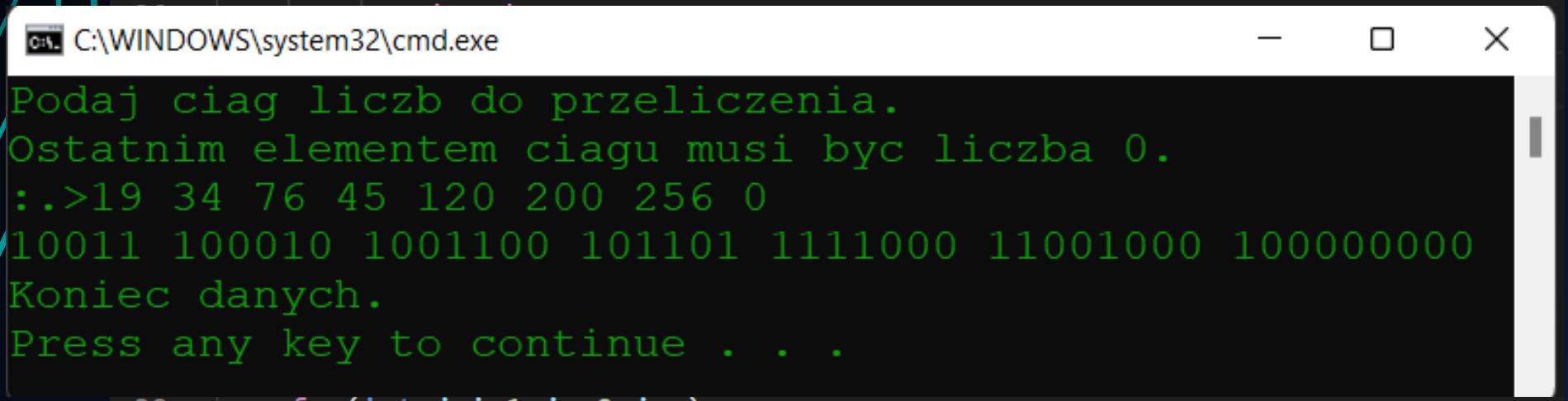
- Napisać program konwertującą podany przez użytkownika ciąg liczb całkowitych zapisanych w systemie dziesiętnym na system szesnastkowy
- Poszczególne liczby w ciągu wejściowym rozdzielone są spacjami
- Ciąg kończy liczba 0
- Zadanie konwersji realizowane jest przez funkcję
- Funkcja potrafi konwertować na jeden z trzech systemów liczbowych

  b-binarny, o-ósemkowy, h-szesnastkowy

Przykład




```
29              dziel=8;
```

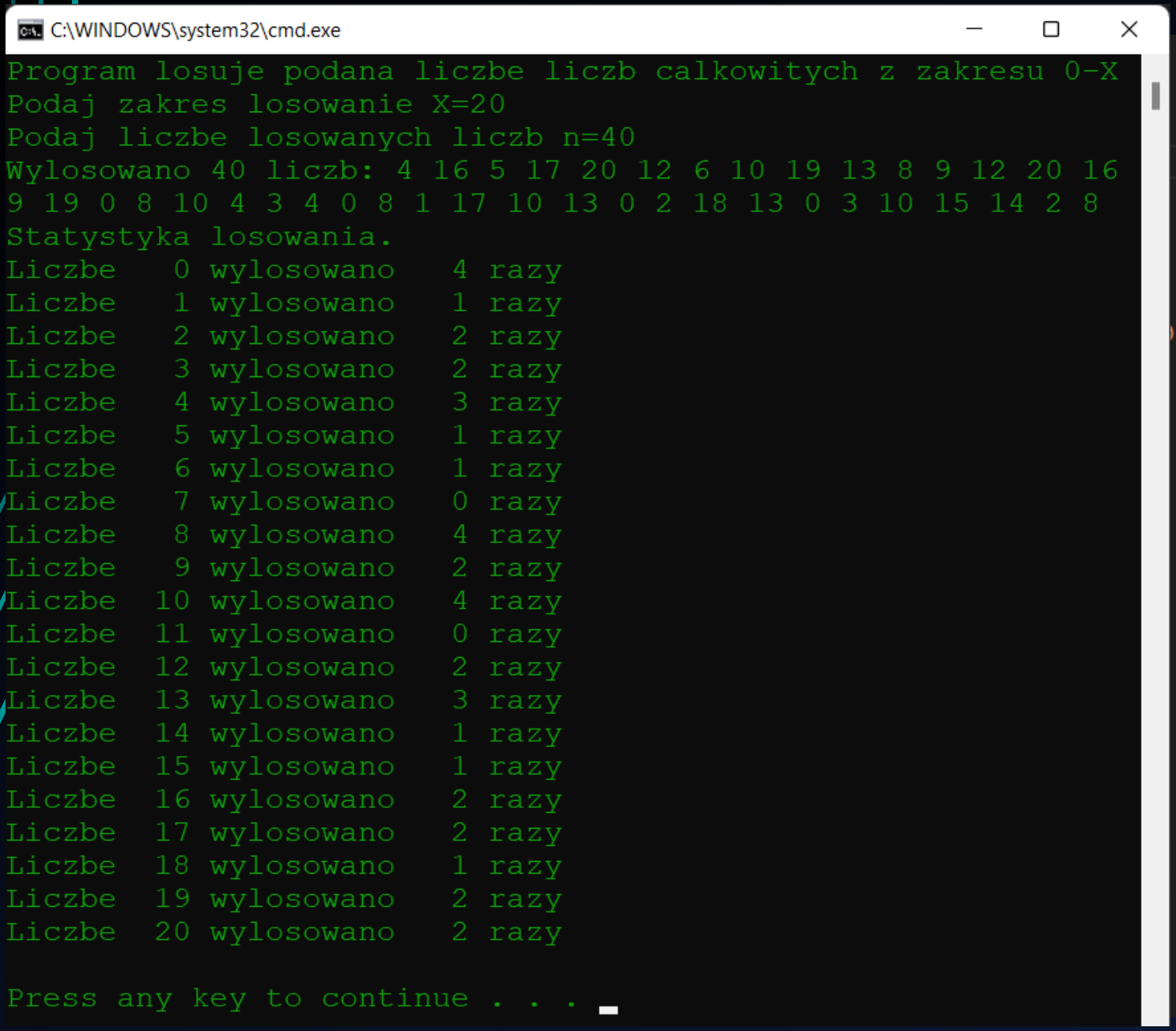



```
39          for(int j=i-1;j>=0;j--)
40              printf("%c",znaki[k[j]]);
41      }
```

Tablice

- Przy deklaracji tablicy określa się jej rozmiar podając liczbę elementów jaką dana tablica będzie przechowywać
- Zdefiniowanie rozmiaru może być realizowane poprzez podanie liczby, inicjację tablicy wektorem elementów lub poprzez zmienną typu `int`
- Parametryzacja rozmiaru pozwala na zadeklarowanie rozmiaru tablicy w sposób dynamiczny, w zależności o bieżących potrzeb aplikacji

Przykład


C:\WINDOWS\system32\cmd.exe
Program losuje podana liczbe liczb calkowitych z zakresu 0-X
Podaj zakres losowanie X=20
Podaj liczbe losowanych liczb n=40
Wylosowano 40 liczb: 4 16 5 17 20 12 6 10 19 13 8 9 12 20 16
9 19 0 8 10 4 3 4 0 8 1 17 10 13 0 2 18 13 0 3 10 15 14 2 8
Statystyka losowania.
Liczbe 0 wylosowano 4 razy
Liczbe 1 wylosowano 1 razy
Liczbe 2 wylosowano 2 razy
Liczbe 3 wylosowano 2 razy
Liczbe 4 wylosowano 3 razy
Liczbe 5 wylosowano 1 razy
Liczbe 6 wylosowano 1 razy
Liczbe 7 wylosowano 0 razy
Liczbe 8 wylosowano 4 razy
Liczbe 9 wylosowano 2 razy
Liczbe 10 wylosowano 4 razy
Liczbe 11 wylosowano 0 razy
Liczbe 12 wylosowano 2 razy
Liczbe 13 wylosowano 3 razy
Liczbe 14 wylosowano 1 razy
Liczbe 15 wylosowano 1 razy
Liczbe 16 wylosowano 2 razy
Liczbe 17 wylosowano 2 razy
Liczbe 18 wylosowano 1 razy
Liczbe 19 wylosowano 2 razy
Liczbe 20 wylosowano 2 razy
Press any key to continue . . .

Tablica a funkcja

- Funkcja nie może zwracać wartości typu tablicowego
- Parametry wejściowe funkcji mogą być typu tablicowego
- Należy pamiętać że tablica deklarowana w języku C jest wskaźnikiem
- Jeżeli tablica jest parametrem wejściowym funkcji to przy wywołaniu przekazuje się do funkcji nie kopię tablicy a wskaźnik do tablicy
- Wszelkie zmiany zawartości elementów tablicy w funkcji powodują także zmiany w zawartości tablicy przekazywanej jako parametr do funkcji

Przykład

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

}
```

```
C:\WINDOWS\system32\cmd.exe
Inicjalizacja tablicy o rozmiarze n.
Podaj rozmiar tablicy n=12
Rozmiar tablicy tablica n=12
Rozmiar tablicy tab wewnatrz funkcji n=1
          indeks:   0   1   2   3   4   5   6   7   8   9  10  11
Zawartosc tablicy:   0   0   0   0   0   0   0   0   0   0   0   0
Losowanie liczb z zakresu 0-11
          indeks:   0   1   2   3   4   5   6   7   8   9  10  11
Zawartosc tablicy:   3   0   1   1   2   1   1   0   0   0   2   1

Press any key to continue . . .
```

```c
    printf("\n");
}                                                    ;
    inicjuj(tablica,n);
    drukuj(tablica,n);
    losuj(tablica,n);
    drukuj(tablica,n);
}
```

Tablica a funkcja

- Jeżeli parametrem wejściowym funkcji jest wskaźnik można za jego pośrednictwem przekazać do funkcji wskaźnik do tablicy

- Jedynie od kodu zdefiniowanego w funkcji zależy jak odwołamy się do podanego adresu

```
void funkcja(int tab[]);
void funkcja(int *tab);
```

Przykład

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

void inicjuj(int*,int);
void losuj(int*,int);
```

C:\WINDOWS\system32\cmd.exe

```
Inicjalizacja tablicy o rozmiarze n.
Podaj rozmiar tablicy n=8
Rozmiar tablicy tablica n=8
Rozmiar tablicy tab wewnatrz funkcji n=1
           indeks:   0   1   2   3   4   5   6   7
Zawartosc tablicy:   0   0   0   0   0   0   0   0
Losowanie liczb z zakresu 0-7
           indeks:   0   1   2   3   4   5   6   7
Zawartosc tablicy:   2   0   1   0   1   2   0   2
           indeks:   0   1   2   3   4   5   6   7
Zawartosc tablicy:   8   8 6422172 6422120   7  11 13112588 6422376

Press any key to continue . . .
```

```c
    drukuj(wsk,n);
}
void inicjuj(int *tab,int m)
{
    int n=sizeof(tab)/sizeof(int);
    printf("Rozmiar tablicy tab wewnatrz funkcji n=%i\n",n);
    for(int i=0;i<m;i++)
        tab[i]=0;
}
```

Alokacja pamięci

- Parametryzacja rozmiaru tablicy nie pozwala na zmianę rozmiaru tablicy w trakcie działania programu – dynamiczne dostosowywanie rozmiaru tablicy
- Efekt ten można osiągnąć za pomocą alokacji (przydzielenia) i relokacji pamięci
- Każda dynamicznie utworzona zmienna musi zostać skasowana
- Dynamiczna alokacja pamięci wymaga podpięcia biblioteki `<stdlib.h>`

Tworzenie zmiennej

- W języku C dostępne są dwie funkcje alokacji (tworzenia) dynamicznej zmiennych

```
*malloc(size_t size);
```

- Funkcja rezerwuje wymagany obszar w pamięci dla zmiennej określonego typu, size określa rozmiar pamięci jaki będzie rezerwowany

```
*calloc(int liczba, size_k size);
```

- Funkcja działa podobnie jak **malloc**, pierwszy parametr określa liczbę komórek pamięci, a drugi rozmiar pojedynczej komórki. Dodatkowo funkcja ustawia wartość każdej komórki na 0
- Wynik działania funkcji zwracany jest do wskaźnika danego typu

Alokacja pamięci

- Dynamiczna alokacja pamięci stosowana jest do tablic.
- Do określania rozmiaru pojedynczego bloku danego typu wygodnie jest stosować funkcję `sizeof(typ);`
- Z racji specyfiki działania funkcja `calloc` jest wolniejsza od funkcji `malloc`
- Lokalnie utworzona tablica nie zostanie skasowana po zakończeniu danego bloku, przydzielony obszar pamięci będzie zablokowany
- Konieczne jest zwolnienie rezerwacji pamięci dla zmiennej

```
free(nazwa);
nazwa = NULL; //opcjonalnie
```

Alokacja pamięci

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

C:\WINDOWS\system32\cmd.exe

```
tab1[0] =     8007400, tab2[0]= 0
tab1[1] =     7995584, tab2[1]= 0
tab1[2] = 1465659955, tab2[2]= 0
tab1[3] = 1868852841, tab2[3]= 0
tab1[4] = 1867543415, tab2[4]= 0
tab1[0] =       -9, tab2[0]=      8
tab1[1] =       -8, tab2[1]=      6
tab1[2] =       -5, tab2[2]=     12
tab1[3] =        0, tab2[3]=     44
tab1[4] =        7, tab2[4]=    120

Press any key to continue . . .
```

```
],i,tab2[i]);
,i,tab2[i]);
```

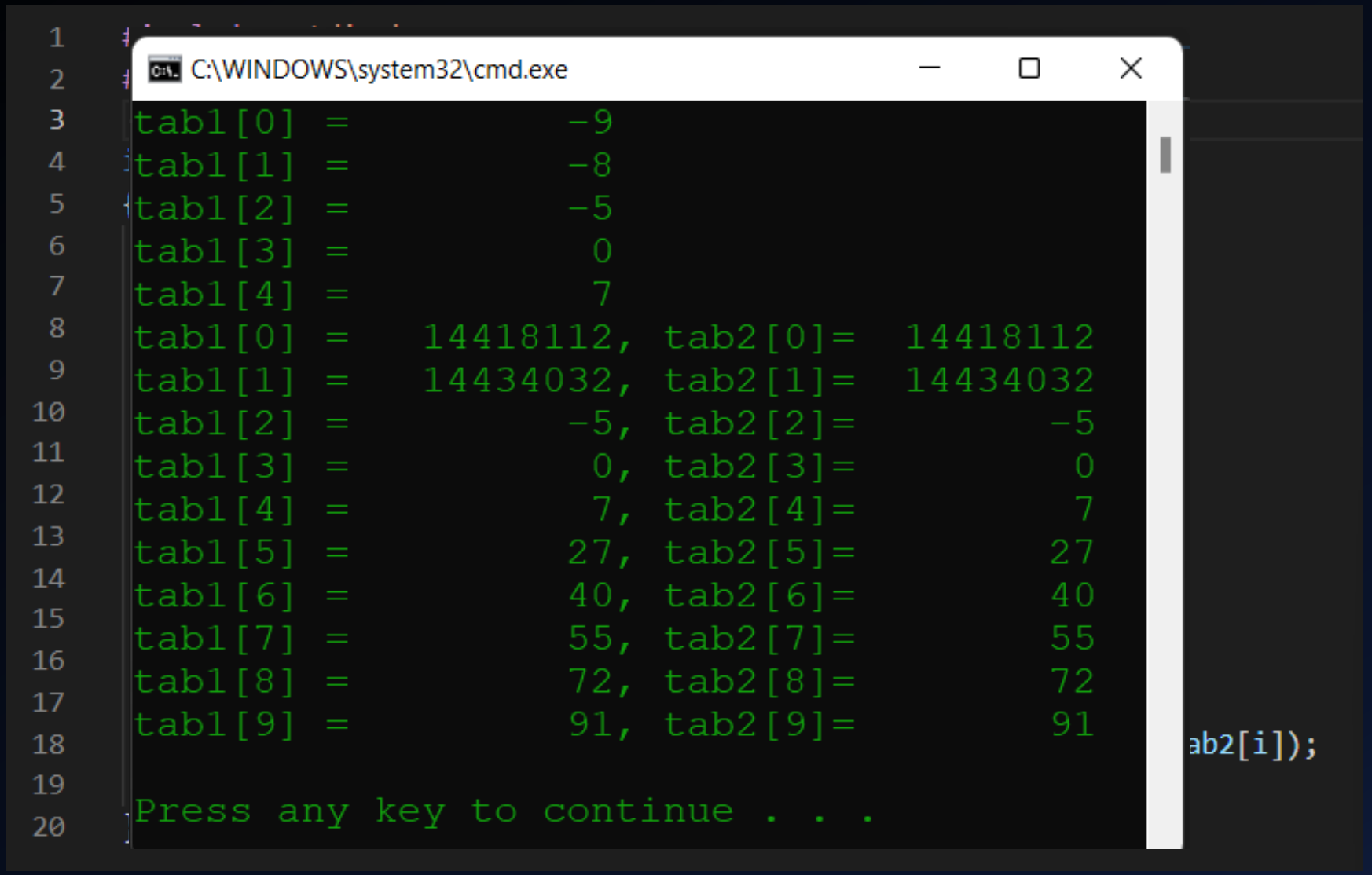
Relokacja

- Gdy konieczna jest zmiana rozmiaru tablicy możliwe jest zmodyfikowanie wielkości przydzielonej dla danej tablicy pamięci

```
*realloc(void *nazwa, size_t size);
```

- Funkcja kopiuje dane z wskazanej tablicy do nowej tablicy o podanym rozmiarze `size`
- Możliwe jest utworzenie tablicy większej lub mniejszej od wejściowej
- Tablica wynikowa przechowuje wskaźnik na tablicę wejściową, ale o zwiększonym rozmiarze
- W funkcji `realloc()` tablica wejściowa i wyjściowa może być ta samą tablicą.

Przykład

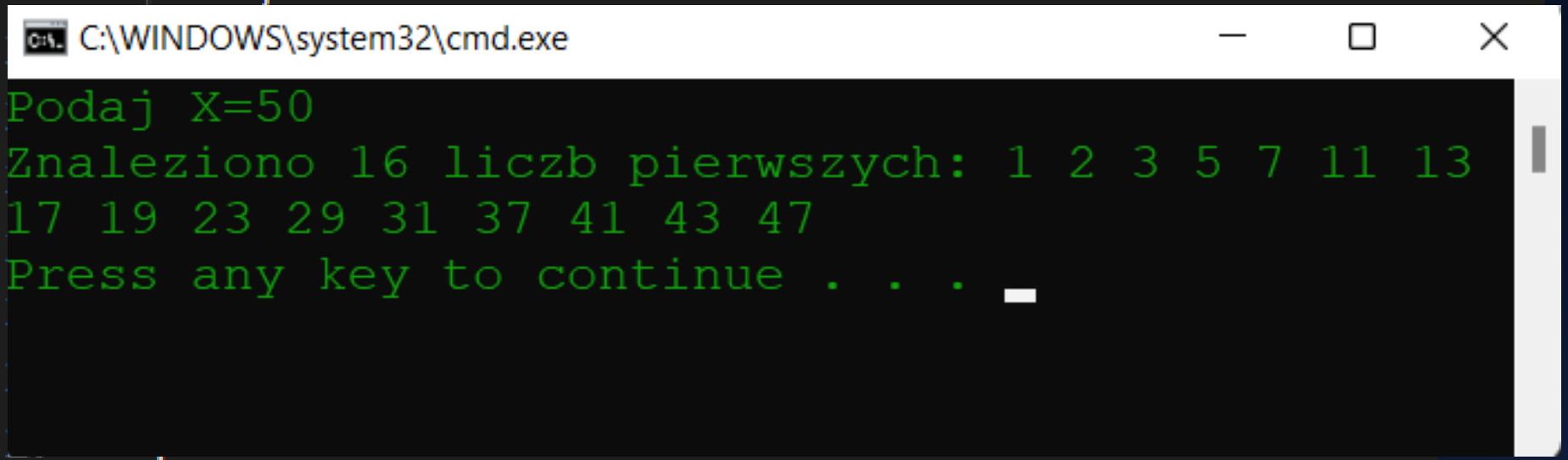

C:\WINDOWS\system32\cmd.exe
tab1[0] = -9
tab1[1] = -8
tab1[2] = -5
tab1[3] = 0
tab1[4] = 7
tab1[0] = 14418112, tab2[0]= 14418112
tab1[1] = 14434032, tab2[1]= 14434032
tab1[2] = -5, tab2[2]= -5
tab1[3] = 0, tab2[3]= 0
tab1[4] = 7, tab2[4]= 7
tab1[5] = 27, tab2[5]= 27
tab1[6] = 40, tab2[6]= 40
tab1[7] = 55, tab2[7]= 55
tab1[8] = 72, tab2[8]= 72
tab1[9] = 91, tab2[9]= 91
Press any key to continue . . .
ab2[i]);

Zadanie

- Napisać program wyznaczający wszystkie liczby pierwsze w przedziale od 0 do X, gdzie X jest dowolną liczbą całkowitą podawaną przez użytkownika
- Program napisać wykorzystując dynamiczną alokację pamięci
- Liczba jest liczbą pierwszą jeżeli jest podzielna tylko przez 1 i samą siebie
- Algorytm sprawdza czy analizowana liczba jest podzielna przez którąkolwiek ze znanych (zapisanych w tablicy) liczb pierwszych. Jeżeli w wyniku dzielenie module nie uzyska się 0 to jest liczbą pierwszą

Zadanie

```
13    while(m<X){
28     printf("Znaleziono %i liczb pierwszych: ",n);
29     for(int i=0;i<n;i++)
30         printf("%i ",liczby[i]);
31     free(liczby);
32     }
```



Tablice wielowymiarowe

- W języku C można deklarować także tablice wyższego stopnia niż jeden
- Każdy następny wymiar określany jest przez dodanie [x] określającą rozmiar danego wymiar
- Najczęściej stosowane są tablice dwuwymiarowe

```
int macierz[10][10];
```

- Pierwsza wielkość odnosi się do liczby wierszy
- Drugi określa liczbę kolumn

Tablice wielowymiarowe

- Przy deklaracji tablicy rezerwowany jest wymagany obszar
- W pamięci rezerwowane są komórki kolejno dla pierwszego wiersza i jego komórek, a następnie dla kolejnych

```
int tab[3][3];
```

		tab[0][0]	tab[0][1]	tab[0][2]
tab[1][0]	tab[1][1]	tab[1][2]	tab[2][0]	tab[2][1]
tab[2][2]				

Tablice wielowymiarowe

- Inicjacja tablicy dwuwymiarowej realizowana jest analogicznie jak jednowymiarowej
- W odróżnieniu od tablicy jednowymiarowej wymagane jest zawsze podanie przynajmniej liczy kolumn (drugi indeks)
- Podanie listy elementów

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

- Podanie niepełnej listy elementów:

```
int macierz[3][3]={{1,2},{2},{3,4}};
```

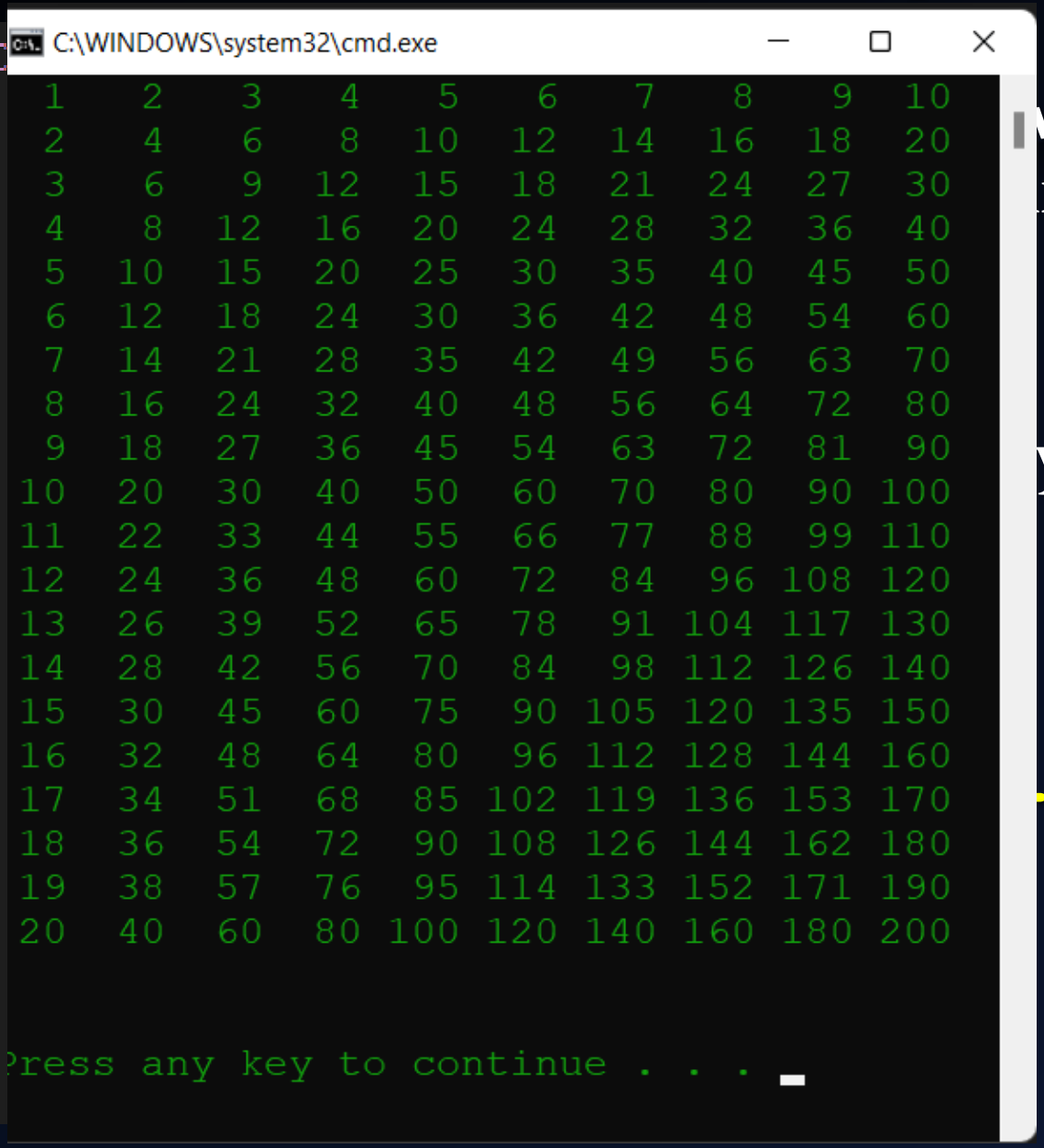
Nieokreślone elementy zostanę zapełnione wartością pustą `0` lub `'/0'`

- Podanie indeksowanej listy elementów

```
int macierz[][6]={[1][1]=4,[5][5]=10};
```

Tablice wielowymiarowe

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



v tablicy
ieżdżenie

y

-)

Funkcje i tablica 2D

- Tablice wielowymiarowe mogą także być przekazywane do funkcji jako parametry

```
void nazwa(int tab [20][20]);
void nazwa(int tab[][20]);
void nazwa(int tab[20][]);
void nazwa(int tab[][]);
```

- Informacje o rozmiarze tablicy można przekazać poprzez parametry

```
void nazwa(int a, int b, int tab[a][b]);
```

- Zaleca się przekazywanie tablicy za pomocą wskaźnika

```
void nazwa(int **tab);
```

Tablice dynamiczne

- Tablice dwuwymiarowe (i większe) mogą być także deklarowane dynamicznie (parametrycznie)

```
int nazwa[n][m];
```

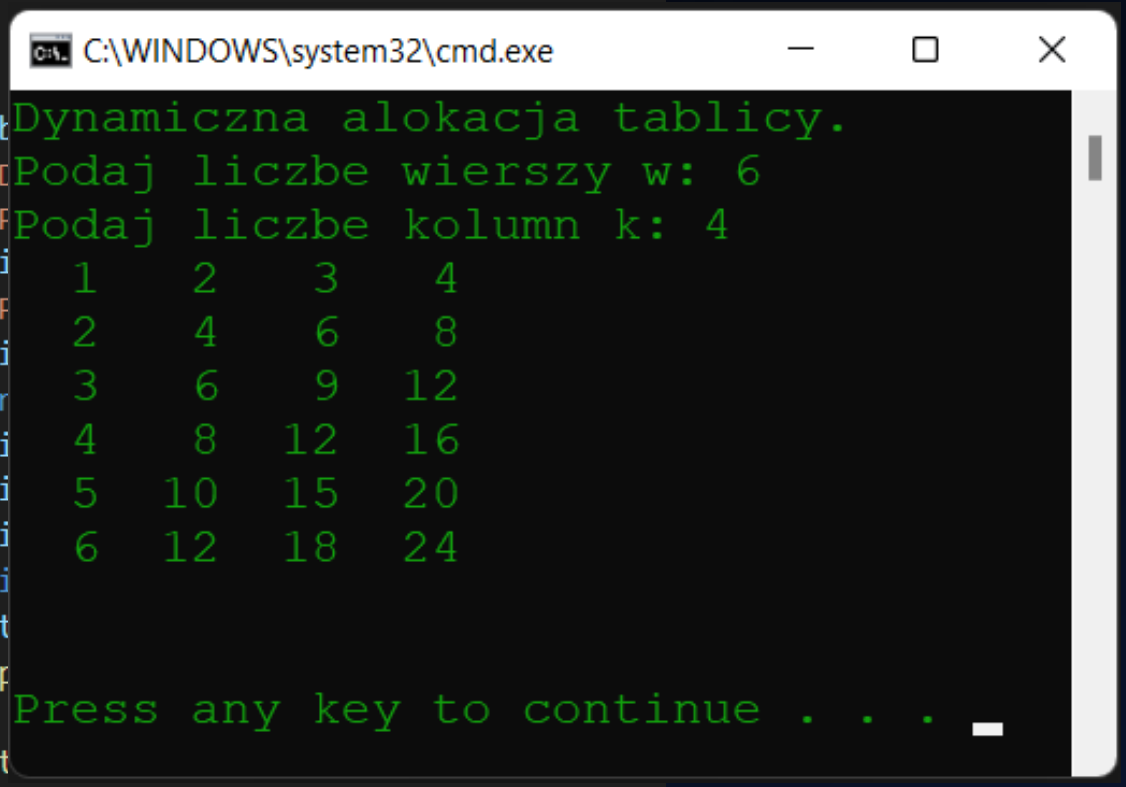
- Nie można inicjować tablic deklarowanych parametrycznie

Alokacja dynamiczna

- Tablice dwuwymiarowe mogą być także alokowany dynamicznie poprzez rezerwowanie pamięci
- W zależności od liczby wymiarów tablicy proces deklaracji i zwalniania wymaga wykonania odpowiedniej liczby operacji
- Dla tablicy dwuwymiarowej najpierw deklaruje się `int **tab;`
- Następnie alokuje się pamięć dla wierszy
- W kolejnym kroku dla każdego wiersza alokuje się wymaganą liczbę komórek
- Przy zwalnianiu pamięci postępuje się w kolejności odwrotnej

Alokacja dynamiczna

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

int main()
{
    int k,w;
    int **tab
    printf("D
    printf("P
    scanf("%i
    printf("P
    scanf("%i
    tab = (in
    for(int i
        tab[i
    for(int i
        for(i
            t
            p
        }
        print
    }
    printf("\n");
    for(int i=0;i<w;i++)
        free(tab[i]);
    free(tab);
}
```



Alokacja dynamiczna

```
C:\WINDOWS\system32\cmd.exe
Podaj liczbe wierszy tablicy 2D: 5
Podaj liczbe komorek w wierszu 0: 3
Podaj liczbe komorek w wierszu 1: 1
Podaj liczbe komorek w wierszu 2: 4
Podaj liczbe komorek w wierszu 3: 2
Podaj liczbe komorek w wierszu 4: 5
TAB[0][0] = bb3f88 TAB[0][1] = bb3f8c TAB[0][2] = bb3f90
TAB[1][0] = bb3fa8
TAB[2][0] = bb3fc8 TAB[2][1] = bb3fcc TAB[2][2] = bb3fd0 TAB[2][3] = bb3fd4
TAB[3][0] = bb15e0 TAB[3][1] = bb15e4
TAB[4][0] = bb1600 TAB[4][1] = bb1604 TAB[4][2] = bb1608 TAB[4][3] = bb160c TAB[4][4] = bb1610

Press any key to continue . . .
```

```
C:\WINDOWS\system32\cmd.exe
Podaj liczbe wierszy tablicy 2D: 5
Podaj liczbe komorek w wierszu 0: 3
Podaj liczbe komorek w wierszu 1: 1
Podaj liczbe komorek w wierszu 2: 4
Podaj liczbe komorek w wierszu 3: 2
Podaj liczbe komorek w wierszu 4: 5
TAB[0][0] =  77 TAB[0][1] =  66 TAB[0][2] =  39
TAB[1][0] =   2
TAB[2][0] =  58 TAB[2][1] =  39 TAB[2][2] =  11 TAB[2][3] =  22
TAB[3][0] =  45 TAB[3][1] =  85
TAB[4][0] =  60 TAB[4][1] =  11 TAB[4][2] =  71 TAB[4][3] =  54 TAB[4][4] =   3

Press any key to continue . . .
```

Relokacja

- Rozmiar tablicy 2D także można modyfikować w podobny sposób jak tablice jednowymiarowe
- Przy zwiększaniu liczby wierszy należy pamiętać o późniejszym zadeklarowaniu (**malloc** lub **calloc**) liczby komórek w nowo utworzonych wierszach

Relokacja

```
29      //relokacja
30      int nw;
31      printf("Podaj nowa liczbe wierszy tablicy 2D: ");
32      scanf("%i", &nw);
33      if(nw<=w){
34          for(int i=nw;i<w;i++)
35              free(tab[i]);
```

C:\WINDOWS\system32\cmd.exe

```
Podaj liczbe wierszy tablicy 2D: 6
Podaj liczbe komorek w wierszu 0: 4
Podaj liczbe komorek w wierszu 1: 2
Podaj liczbe komorek w wierszu 2: 1
Podaj liczbe komorek w wierszu 3: 5
Podaj liczbe komorek w wierszu 4: 4
Podaj liczbe komorek w wierszu 5: 3
TAB[0][0] =  22 TAB[0][1] =   3 TAB[0][2] =  41 TAB[0][3] =  33
TAB[1][0] =   7 TAB[1][1] =  38
TAB[2][0] =  52
TAB[3][0] =  37 TAB[3][1] =  31 TAB[3][2] =  61 TAB[3][3] =  45 TAB[3][4] =  45
TAB[4][0] =  87 TAB[4][1] =  85 TAB[4][2] =  39 TAB[4][3] =  28
TAB[5][0] =  77 TAB[5][1] =  85 TAB[5][2] =  53
Podaj nowa liczbe wierszy tablicy 2D: 3
TAB[0][0] =  22 TAB[0][1] =   3 TAB[0][2] =  41 TAB[0][3] =  33
TAB[1][0] =   7 TAB[1][1] =  38
TAB[2][0] =  52

Press any key to continue . . .
```

```
56      free(k);
57      for(int i=0; i<nw; i++)
58          free(tab[i]);
59      free(tab);
60  }
```