МАСИВИ

Масиви належать до складених (агрегованих) типів даних.

Масив – це набір даних однакового типу, об'єднаних загальним іменем та розташованих поспіль у оперативній пам'яті.

Основні властивості, якими характеризуються масиви мови C:

- всі елементи масиву мають однаковий тип і спільне ім'я;
- масив займає неперервну ділянку оперативної пам'яті;
- елементи масиву розташовуються послідовно за порядком зростання індексів;
- індекс першого елемента масиву завжди дорівнює `0`;
- до елементів масиву можна звертатись через індекси або через вказівники;
- ім'я масиву є вказівником-константою на його початок в оперативній пам'яті, тобто ім'я масиву зберігає адресу першого елемента цього масиву;
- масив може бути одновимірним чи багатовимірним (масив, сформований з масивів).

Перелічені властивості є основою для опрацювання масивів у C-програмах та організації звертання до їхніх елементів.

Оголошення та ініціалізація одновимірних масивів

Як і звичайні змінні, всі масиви мають бути оголошені явно, щоб компілятор міг виділити для кожного з них ділянку пам'яті відповідного розміру.

Оголошення масивів виконують такою конструкцією:

тип_елементів ім'я_масиву `[`***кількість_елементів***`];`

де ***тип_елементів*** може бути довільним допустимим для C простим чи складеним типом;
ім'я_масиву – ідентифікатор, що відповідає правилам запису імен;
квадратні дужки `[]` – обов'язкова ознака масиву;
кількість_елементів – константа чи константний вираз, що визначає розмірність масиву.

Приклади оголошень масивів:

```
double arr[15];                    /* масив з 15-ти дійсних чисел */
int vector[4*N];            /* масив з 4*N цілих чисел, N-константа */
```

Для кожного масиву виділяється неперервна ділянка пам’яті, розмір якої дорівнює:

кількість_елементів `×` `sizeof(`***тип_елементів***`)`.

Елементи масиву завжди розташовуються в оперативній пам’яті один за одним. Найменшу адресу має перший елемент (тобто, елемент з індексом `0`), а найстаршу – останній елемент масиву. Індекс останнього елемента масиву завжди на 1 менший за кількість елементів цього масиву. Рисунок нижче ілюструє, як будуть розташовані у пам’яті 15 елементів масиву `arr`, що мають тип `double`.

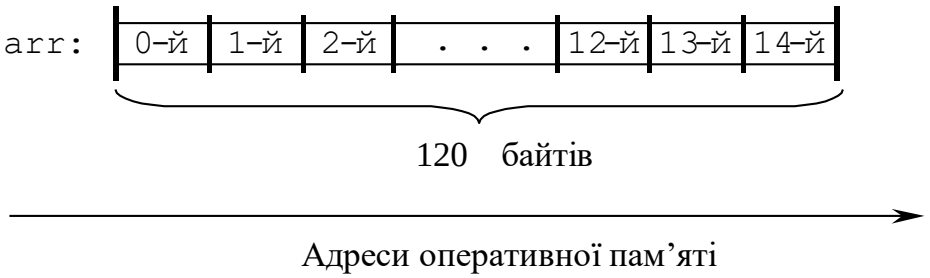

Рис. Порядок розташування елементів масиву `arr`

Оголошуючи масиви, можна відразу ініціалізувати їхні елементи. Застосовують дві форми ініціалізації:

- з вказанням кількості елементів масиву;
- без зазначення розмірності масиву.

У першому випадку, після розмірності масиву у фігурних дужках `{ }` послідовно записують значення елементів, починаючи від першого (з індексом `0`).

```
int prime1[5] = {2, 3, 5, 7, 11};
```

Кількість заданих значень може бути меншою за кількість елементів масиву:

```
short prime2[50] = {2, 3, 5};
```

– значення отримають тільки перші три з 50-ти елементів масиву `prime2`, решта елементів залишаться невизначеними.

Якщо ініціалізуються всі елементи масиву, то ***кількість_елементів*** можна не вказувати. Розмірність масиву буде визначатись кількістю елементів-ініціалізаторів:

```
char golosni[] = {'а', 'о', 'е', 'и', 'і', 'у', 'є', 'ї', 'я'};
```

Масив `golosni` буде складатися з 9-ти елементів з типом `char`.

Розмірність масиву, оголошеного без вказання кількості елементів, можна обчислити в програмі, використовуючи операцію `sizeof`:

кількість_елементів_масиву = `sizeof(`*ім'я_масиву*`) / sizeof(`*тип_елементів*`)`

Кількість елементів масиву `golosni`:

```
sizeof(golosni) / sizeof(char)
```

Операція `sizeof(`*ім'я_масиву*`)` повертає розмір усієї ділянки, яку займає масив, незалежно від того, наскільки він заповнений:

`sizeof(arr)` ⟶ `120` (15 елементів по 8 байтів);

`sizeof(prime2)` ⟶ `100` (50 елементів по 2 байти).

Звертання до елементів масиву через індекси

Елемент масиву є змінною (точніше – первинним виразом) з типом, заданим в оголошенні масиву. У програмі елемент масиву синтаксично може записуватися всюди, де потрібне дане такого типу.

Для доступу до елементів масиву використовують дві форми звертання:

1) через індекси;
2) через вказівники (адреси).

Для звертання через індекси застосовують конструкцію:

ім'я_масиву `[`*індекс_елемента*`]`

де *індекс_елемента* довільний цілочисловий вираз, що задає порядковий номер (індекс) елемента в масиві – нумерація індексів починається з `0`.

Приклади:

`prime2[0]` – перший елемент масиву `prime2`;

`golosni[4]` – п'ятий за порядком елемент масиву `golosni`;

`arr[2*k-5]` – елемент масиву `arr`, індекс якого дорівнює значенню виразу `2*k-5`.

Важливо – компілятор C не контролює відповідність індексів до розмірності масиву.

Тому в критичних точках програми доцільно виконувати перевірку правильності значень індексів, наприклад:

```
if (i>=0 && i<NEL)          /* NEL – розмірність масиву ar */
    ar[i] = ... ;
```

Мова С не виконує ніяких операцій з цілими масивами. Всі операції, зокрема введення-виведення, виконуються поелементно.

Приклад фрагмента програми, де вводяться елементи масиву беззнакових чисел. Введення припиняється, коли зчитано **MAX** чисел або введено нечисловий символ.

```
/*  Введення елементів масиву  */
#include <stdio.h>
#define MAX 100                    // максимальна к-сть елементів
          . . .
        unsigned mas[MAX];
        int i, k;                  // k – кількість введених елементів
    printf(" Введіть не більше %d елементів масиву "
           "(довільна літера для завершення): \n", MAX);
    for( k=0;  k< MAX; k++ )
       if  ( scanf("%u", &mas[k])==0 )     // введення k-го елемента
          break;
```

Приклад програми, яка зі заданого масиву дійсних чисел створює новий масив.

```
/*  Формування нового масиву, кожен елемент якого дорівнює
    півсумі сусідніх елементів базового масиву  */
#include <stdio.h>
int  main(void)
{
  double base[] = {0.34, 14.68, 123.8, 21.78, 90.07, 55.6, 12.07};
  const int nsize = sizeof(base)/sizeof(double)-1;
  double nar[nsize];                    // новий масив розміром nsize
  int i;
   for(i = 0;  i < nsize;  i++)         // формування масиву nar
      nar[i] = (base[i]+base[i+1])/2;
   printf("\n\t\t  Новий масив: \n");
   for(i=0;  i<nsize; i++)              // друк нового масиву
      printf("%7.2lf", nar[i]);
   return 0;
}
```

Результат виконання:

```
          Новий масив:

  7.51   69.24   72.79   55.93   72.83   33.84
```