Тема 9
Дефиниране и използване на масиви в език C/C++

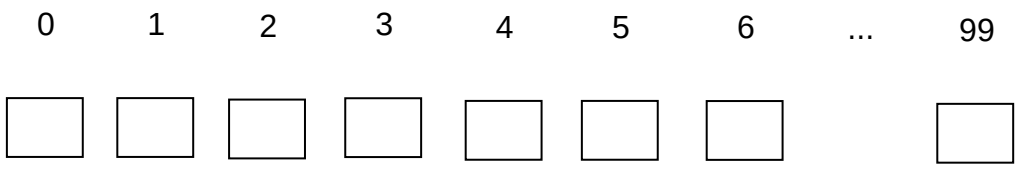
1. Обща характеристика на тип данни *масив*

Типовете данни в програмните езици се разделят най-общо на ***прости (скаларни)*** и ***структурирани***. Скаларните типове данни са тези, които се състоят от един елемент. В C/C++, пример за такива данни са аритметичните типове: ***int***, ***char***, ***float*** и ***double***. Структурираните типове се състоят от повече от един елемент.

Масивът е структура от данни, състояща се от множество последователно наредени елементи от един и същи тип, достъпът до който се осъществява чрез името на масива и поредния номер на елемента т.нар. **индекс**.Типът **масив** се характеризира със следните особености:

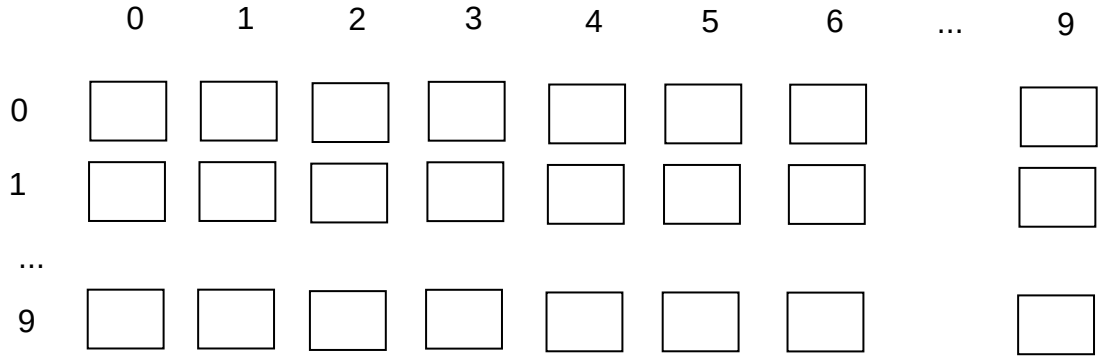
- *структуриран тип данни от еднотипни елементи* - за разлика от простите типове, масивите се състоят от повече от един елемент, като елементите задължително са от един и същи тип. Типът на елемента определя типа на масива.
- *структуриран тип данни от статичен вид* - броят на елементите на масива трябва да се определи по време на компилация; той не се изменя в процеса на изпълнение на програмата. Компилаторът заделя необходимата памет.
- *индексиран тип данни* - достъпът до даден елемент се осъществява чрез името на масива и неговият ***индекс*** (пореден номер).

Масивите, според размерността си се разделят на: ***едномерни*** и ***многомерни***. Многомерните масиви се разделят на двумерни, тримерни и т.н. Едномерните масиви могат да се представят като редица от последователно наредени елементи (фиг. 9.1). Двумерните се представят като матрици (фиг. 9.2). В тримерното пространство е трудно да се даде представа за масив с повече от три индекса и затова такъв тип масиви рядко се използва.



Фиг. 9.1. Структура на едномерен масив от 100 елемента

В паметта на компютъра многомерните масиви се представят като едномерни. Например, двумерен масив се нарежда последователно ред по ред. Двумерният масив се третира като *едномерен масив, чиито елементи са едномерни масиви*.



Фиг. 9.2. Структура на двумерен масив от 10x10 елемента

2. Дефиниране на масиви. Достъп до елементите

Общият вид на дефиницията на едномерен масив е следния:

имеТип имеМасив[размерност];

Съответно:

имеТип - определя типа на елементите на масива; може да е някой от стандартните типове или дефиниран от програмиста;

имеМасив е *идентификатор*, определящ името на променливата от тип *масив*;

размерност - константен израз, който определя броя на елементите в масива и се нарича още ***граница***.

Примери за дефиниции на масиви са:

```
int massiv[10];    /* дефиниране на едномерен масив massiv от 10 цели елемента */

char str[20];    // дефиниране на едномерен масив ch от 20 символа

float a[10], b[20];  /* дефиниране на два едномерни, реални масива a и b от 10 и 20, елемента съответно */
```

Достъпът до даден елемент на масива е чрез името на масива и индекса на елемента. Например:

```
massiv[0] = 10;

a[2]=0.5;

b[18]=1.9;

str[10]='y';
```

Особеност на език C/C++ относно масивите е, че индексите на елементите започват от **0** (нула), т.е. ако даден масив е с **n** елемента, първият елемент е с номер **0**, а последният е с номер **n-1**. Например, масив **a** от 10 елемента има следните индекси :

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

Дефинирането на многомерни масиви е чрез следната декларация:

име_тип име_масив[граница1][граница2].....;

Примери:

```
float ax[10][20];  /* дефиниране на двумерен масив ax от 10x20 реални елемента  */
int bx [10][10][5];      /* дефиниране на тримерен масив bx от 10x10x5 целочислени елемента  */
```

Индексите при многомерните масиви се изменят по същия начин като едномерните: от **0** до **n-1**.

Двумерният масив е частен случай на многомерен. Достъпът до елементите е чрез два индекса. Подобно на математическото описание на матрица, първият индекс е за редове, вторият за стълбове.

Примери за достъп до елементи на многомерни масиви са:

```
ax[0][0]=0;
bx[0][0][0]=1;
```

Относно разположението на многомерните масиви в паметта на компютъра, елементите се подреждат последователно, по реда на нарастване на индексите, като най-бързо нарастват десните индекси.

Например, подредбата на матрица ***ax*** в паметта на компютъра е:

ax[0][0] ax[0][1] ax[0][2]……..ax[0][19] ax[1,0]……

3. Примерни задачи за работа с масиви

3.1. Едномерни масиви

За обработката на масиви се използват итерационните конструкции и в програмен език C/C++ това най-често е конструкция ***for***, чрез която се обхожда масива от първия до последния елемент или обратно.

Пример 1. Въвеждане на елементите на едномерен масив от целочислени елементи от клавиатурата и извеждане на стойностите в обратен ред.

Задачата е пример за обхождане елементите на масива от първия до последния и обратно – от последния до първия.

```
#include <stdio.h>
main()
{
      int mA[10],i;
      for(i=0; i<10; i++)     //въвеждане стойности на елементите
      {
            printf("Item[%d]=",i);
            scanf("%d",&mA[i]);
      }

      printf("\n");           // извеждане стойностите на елементите
      for(i=9; i>=0; i--) printf("%d ",mA[i]);
```

```
	return 0;
}
```

Пример 2. Търсене на средно аритметична стойност.

Задачата е да се намери средноаритметичната стойност от елементите на едномерен масив от 10 реални елемента с двойна точност. В реализацията, при намирането на общата сума на елементите, чрез конструкция ***for*** се обхожда масивът и при всяка итерация към общата сума се добавя стойността на поредния елемент.

```
#include <stdio.h>
main()
{

	double mB[10];
	int i;
	double suma, average;
	for(i=0;i<10;i++)	//въвеждане стойностите на елементите
	{
		printf("Item[%d]=",i);
		scanf("%lf",&mB[i]);
	}

	suma = 0;		// намиране сумата на елементите
	for(i=0; i<10; i++) suma += mB[i];
	average = suma/10;	// намиране на средноаритметична стойност
	printf("\nAverage = %lf",average);

	return 0;
}
```

3.2. Двумерни масиви

При работата с двумерни масиви е необходимо да се използват вложени цикли. Обикновено итерациите се реализират с две вложени една в друга конструкции ***for***.

Пример 1. Въвеждане и извеждане елементите на друмерен масив.

Задачата е да се въведат стойностите на двумерен масив от 5x4 целочислени елемента и да се изведат на екрана в подходящ фомат. Примерното решение е следното:

```
#include <stdio.h>
main()
{

	int matrix[5][4];
	int i,j;

	for(i=0;i<5;i++)
	{
		for(j=0;j<4;j++)
		{
			printf("Item[%d][%d]=",i,j);
			scanf("%d",&matrix[i][j]);
		}
	}
```

```
	for(i=0;i<5;i++)
	{
		printf("\n");
		for(j=0;j<4;j++)
		{
			printf("%d ",matrix[i][j]);
		}
	}
	return 0;
}
```

4. Инициализиране на масиви

При дефинирането на масиви могат да бъдат зададени начални стойности на елементите, т.е. *масивите да бъдат инициализирани*. Определянето на тип масив, заедно със задаване на начални стойности на елементите е чрез следната дефиниция:

име_тип име_масив [размерност]={списък стойности};

Стойностите на елементите се разделят със запетаи. Пример:

```
int array[3] = {1, 3 , 5};
```

Съответно стойностите на елементите са: array[0]=1; array[1]=3; array[2]=5.

Възможно е, когато се инициализира масив, да не се задава брой на елементите. Тази стойност се определя от броя на зададените стойности. Пример:

```
double d[] = {2.23, 0, 18.5, 4.8};
```

Масивът d е от четири реални елемента с двойна точност; стойностите на елементите са: d[0]=2.23; d[1]=0; d[2]=18.5; d[3]=4.8

Ако размерът на масива е по-голям от броя на зададените стойности, инициализират се първите елементи с посочените стойности, а останалите приемат стойност **0** (нула). Пример:

```
double d[10] = {2.23, 0, 18.5, 4.8};
```

В случая, задават се стойности само на първите четири елемента, а останалите шест приемат стойност 0.

При инициализацията на двумерни масиви се изисква да се зададе множество от списъци, съответстващи на редовете на двумерният масив.

Пример:

```
int array[2][3] = { {1,1,0}, {0,1,1}
                  };
```

Относно въпроси по темата на адрес: ln_zh_st@yahoo.com