Упражнение 4 - Масиви

Понятие за масив

Масивът предоставя възможността група от променливи от един тип да бъдат обединени от едно име, като достъпът до този елемент се осъществява чрез индекс. По този начин данни като дискретни отчети от данни могат да бъдат представяни в паметта.

Дефиниране на масиви в С

В програмният език С масивите се използват чрез:

```
<тип> <име_на_масив>[<размер>];
```

Тук **тип** не се различава по значение от типовете данни, които се използват за дефинирането на променливи. и могат да бъдат използвани всички познати типове от данни.

Масивът като всяка друга променлива изисква да има **име**. Това име е нужно, за да може да се осъществи достъп до елементите на масива.

По време на дефинирането на масив е необходимо да се зададе **размерът** – броят на елементите. Размерът представлява положително константно число и размерът не може да бъде променян повреме на изпълнението на програмата.

Пример за дефиниране на масив:

```
float measurements[100];
```

Начални стойности на масив

В програмният език С елементите на масива нямат начална стойност по под разбиране. Те приемат стойностите на клетките в паметта преди заделянето им от масива.

Начални стойности могат да бъдат заделени чрез изреждане на началните стойности по следния начин:

```
float measurements[5]={1.3, 1.5, 1.6, 1.8, 1.9};
```

Когато броят на началните стойности е по-малък от броят на всички елементи, то останалите елементи приемат начална стойност 0;

В случаите, когато броят на началните стойности е по-голям от размера на масива, то излишните елементи се игнорират;

Достъп до елементите на масив

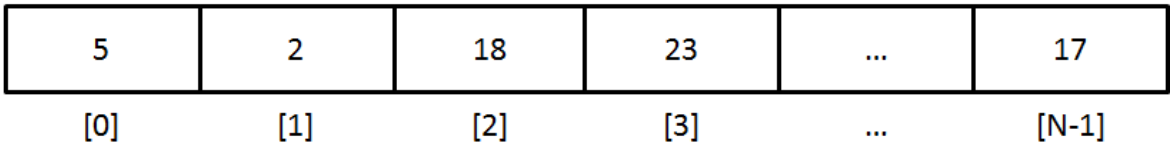
Работата с елементите на масива не се различава като начин от работата с обикновени променливи, но тук се изисква допълнително индекс на съответния елемент в масива.

хон.ас. инж. Боян Петров, ***b_petrov@tu-sofia.bg***,
София, 2013

Индексът представлява цяло неотрицателно число, което съответства на позицията на съответния елемент в масива. В програмният език C достъпът до първия елемент на масива се осъществява чрез индекс 0 (Masiv[0]). Последният елемент в масива има индекс **размер-1**. За начинаещи програмисти това често е трудност в разбирането и често срещана тяхна грешка при обхождането на масива. Идеята тук е идентична на проблема, разгледан в първото упражнение: Броят на цифрите в десетична бройна система е 10, най-голямата цифра 9.

Представяне на елементите на масива в паметта

Masiv_A:



Нека Masiv_A е целочислен масив с N елемента. Разположението на елементите на в паметта е в съседни клетки. Като **позицията** на **първия елемент** е кръстена Masiv_A. Позицията представлява адресът на тази клетка в паметта. Операторът [] показва, че ще бъде достъпена **стойността** на елемент, който се намира на отстояние определено число от началната позиция на масива. Поради тази причина началният елемент има индекс, равен на 0, защото се намира на нулево отстояние от началната позиция, която е посочена от името на масива.

Работа с масиви

За работата с масиви се използват цикли за достъп до елементите на масива, където броячът представлява индекс на съответния елемент. В следващият пример е показано въвеждането на стойностите от масива от клавиатурата и извеждането им в обратен ред на екрана:

```c
#include <stdio.h>

int main()
{
	int i;
	int array[10];
	for(i = 0; i < 10; i++)
	{
		printf("Въведете стойност за елемент %d : ",i);
		scanf("%d",&array[i]);
	}
	printf("Извеждане на елементите на масива в обратен ред:\n");
	for(i = 10-1; i>=0; i--)
	{
		printf("Елемент %d има стойност: %d\n",i,array[i]);
	}
	return 0;
}
```

Символни масиви – низове

Низовете (стрингове) преставляват специален тип масиви. Те се използват за съхранението на текст, представен като последователност от символи. Както беше разгледано в упражнение 1, в програмният език С съществува тип от данни, който може да съхранява символи, следователно масив от символи ще наричаме символен низ или стринг.

При използване на стрингове е важно да се отбележат няколко важни точки:

- Символните низове в С са така наречените „null terminating" това означава, че **задължително** последният елемент на масива трябва да има стойност 0 ('\0'). В противен случай може да настъпи нерегламентиран достъп до полета от паметта, които са извън заделените за този низ.
- Константите низове могат да се представят като текст, заграден от двойни кавички: "Tekst". Така записан константния низ е с последен елемент 0, която се задава неявно.
- С такъв константен низ може да бъде зададена начална стойност на дефиниран масив от символи, като тази операция е разрешена само на това място:

```
char string1[16] = "Some text here!"; //Правилно
char string2[16];
string2 = "Any text there!"; //Неправилно
```

- За операции като присвояване, слепване, сравнение, взимане на размер се използват функции от **string.h**
    - strcpy(<име на низ>, <нова стойност>); - задава стойност на низ;
    - strcat(<низ 1>, <низ 2>); - слепва два масива, като след края на първия масив започва се добава стойността на втория масив;
    - strcmp(<низ 1>, <низ 2>); - лексикографски сравнява два низа, като връща 0, ако двата низа са еднакви. Ако са различни, върната стойност е разликата на първите различаващи се символи;
    - strlen(<низ>); - връща големината на низа, като този размер не размерът заделен в паметта, а броят на символите, записани в този масив;
- Въвеждането на масиви от клавиатурата:

```
scanf("%s", string);
```

Само при въвеждането на низове от клавиатурата не се налага използването на &, защото както беше споменато името на масива носи информация за позицията на първия му елемент .

Сортиране на масиви

Сортирането на масивите представлява подреждане на елементите на масива по даден критерии, като често този критерии е поголемина във възходящ или низходящ ред. Съществуват много алгоритми за сортиране на масиви, като те главно се делят на два вида: бързи и бавни. Разликата между тези два типа е броят стъпки, които са нужни за сортирането на масива. Предимсвата на бавните алгоритмни са, че лесно се имплементират, но броят на стъпките е от в квадратична зависимост броят на елементите, които трябва да бъдат сортирани.

Метод на пряката селекция

Идея на алгоритъма

Алгоритъмът намира оптималния елемент (минимален/максимален) от масива и го поставя в началото на масива, а старото начало се поставя вътре в масива. След това търси оптимална стойност от останалите стойности на масива като намерената стойност отново се поставя в началото на подмасива. Стъпките се повтарят до пълното сортиране на масива.

Реализация

```
#include <stdio.h>

int main()
{
	int i,j, n, min_index;
	int temp, array[100] = {2,7,4,9,1,7,2};
	n = 7; //В текущия пример ще бъдат сортирани първите 7 елемента на масива
	for(i = 0; i<n-1; i++)
	{
		min_index = i; //Първоначално допускане, че първият елемент на масива е с минимална стойност
		//Търсене на елемент с по-ниска стойност, като се запазва индексът на тази елемент
		for(j = i+1; j < n; j++)
			//Проверяване дали съществува елемент с по-малкастойност из следващите елементи на масива
			if(array[min_index] > array [j])  //Условие за размяна
				min_index = j;
		// Разменяне на стойностите
		temp = array[min_index];
		array[min_index] = array[i];
		array[i] = temp;
	}
	return 0;
}
```

„Метод на мехурчето“

Идея на алгоритъма

При този алгоритъм се сравняват всеки два съседни елемента на масива, ако не е правилно тяхното местоположение, то те сменят ствоите позиции. При този алгоритъм се наблюдава ефекта, че по-леките елементи „изплуват“, а по-тежките „потъват“.

Реализация

```
#include <stdio.h>

int main()
{
	int i,j, n;
	int temp, array[100] = {2,7,4,9,1,7,2};
	n = 7;
	for(i = 0; i<n; i++)
	{
		for(j = 0; j < n-i-1; j++)
			if(array[j] > array[j+1])
			{
				temp = array[j];
				array[j] = array[j+1];
				array[j+1] = temp;
			}
	}
	return 0;
}
```

Метод на непосредственото разместване

Идея на алгоритъма

Този алгоритъм се явява като комбинация от идеите на двата алгоритъма, описани по-горе. В този алгоритъм за всеки елемент на масива с индекс i се търси елемент с оптимална стойност (минумум или максимум) из между елементите до края (по идеята на метода на пряката селекция). При намиране на такъв елемент стойностите се разместват веднага (по идеята на „метода на мехурчето“)

Реализация

```
#include <stdio.h>
int main(){
	int i,j, n;
	int temp, array[100] = {2,7,4,9,1,7,2};
	n = 7;
	for(i = 0; i<n-1; i++)
	{
		for(j = i+1; j < n; j++)
			if(array[i] > array[j])
			{
				temp = array[j];
				array[j] = array[i];
				array[i] = temp;
			}
	}
	return 0;
}
```

Задачи за изпълнение

1. Да се напише програма, която в масив до 100 елемента намира най-малкият по големина. Размерността на масива и стойностите на елементите се въвеждат от клавиатурата.
2. Да се напише приграма, която намира на масив с до 100 елемента средно аритеметично на елементите и извежда всички елементи по-големи от тази стойност на екрана. Размерността на масива и стойностите на елементите се въвеждат от клавиатурата.
3. Да се напише програма, в която се въвежда стринг до 30 символа и се преобразува в главни букви.
4. Да се програма, която въвежда цяло положително число от клавиатурата и го преобразува в символен низ, като всеки елемент на символния низ представлява една цифра от числото.
5. Да се напише програма, която преобразува символен низ в цяло число и го умножава го по 2 и го извежда на екрана.