Масиви

Arrays

Масиви

- Елементи от един и същи тип данни подредени последователно в паметта
- Декларация
  - тип_данни име[брой_елементи];
  - Броят на елементите е задължително константна целочислена стойност
    - Число

    float mas[10];
    - Константна променлива

    const int n = 10;

    float mas[n];
    - Макрос за константа

    #define N 10

    float mas[N]

Достъп до елементите на масив

```
#define N 6
int mas[N];
```

индекси	0	1	2	3	4	5
mas	**3**	**-2**	**1**	**4**	**0**	**7**

- Чрез името на масива и индекс на елемента – mas[index]
- Индексът задължително е в интервала [0; N-1]
- Елемент от масива се използва по същия начин като всяка променлива
  - Участие в изрази
  - Задаване на стойност
  - Извеждане
  - и др.
- mas[1] – връща стойността на елемента от масива с индекс 1
- mas[3] = -5; - записва -5 в елемент с индекс 3

	0	1	2	3	4	5
mas	**3**	**-2**	**1**	**-5**	**0**	**7**

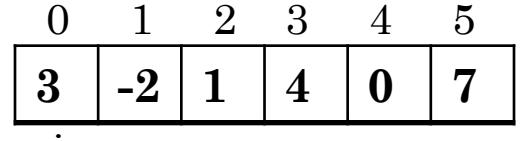
Обхождане на масив

- Обхождането означава преминаване през всички елементи от масива
- Обикновено се използва цикъл, като всяка итерация на цикъла се обработва един елемент от масива
- Като индекс при достъп до елемент от масив може да се използва променлива – стойността на променливата дава индекса на елемента
  - mas[i], при i = 0 – елемент с индекс 0

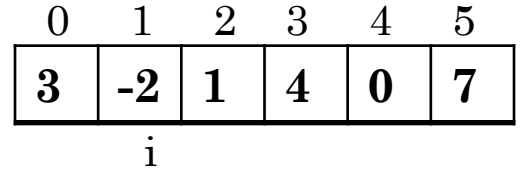
Обхождане на масив – цикъл for

```
for(i = 0; i < n; i++)
{
Обработка на mas[i];
}
```

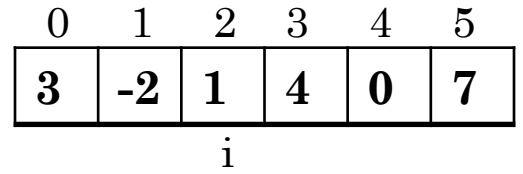
1 итерация) i = 0, mas[i] → mas[0]



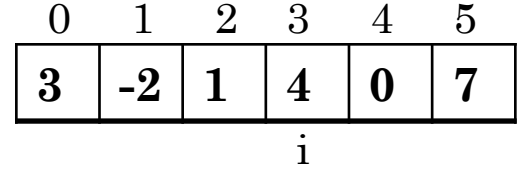
2 итерация) i = 1, mas[i] → mas[1]



3 итерация) i = 2, mas[i] → mas[2]



4 итерация) i = 3, mas[i] → mas[3]



.

.

.

Обработка

```
const int n = 5;
int mas[n];
```

- Въвеждане на стойности за елементите на масив

```
for(i = 0; i < n; i++)
{
        printf("Vavedete stoinost za %d element, i+1); //i+1, за да изведе 1 element, а не 0
        scanf("%d", &mas[i]);
}
```

- Извеждане на стойностите на елементите на масив

```
for(i = 0; i < n; i++)
{
        printf("element[%d] = %d", i, mas[i]);
}
```

- Намиране на сумата на елементите

```
for(i = 0; i < n; i++)
{
        sum += mas[i];
}
```

Намиране на минимална стойност и нейния индекс

```c
#include <stdio.h>

#define N 100

int main()
{
  float mas[N], min;
  int n, ix, minIx;

  printf("Kolko stoinosti ste vavedete:");
  scanf("%d", &n); //в масива има 100 елемента, но можем да работим и само с част от тях, колкото са необходими

  for (ix = 0; ix < n; ix++) //при обхождането на масива ограничаваме до необходимия брой елементи
  {
    printf("Vavedete %d stoinost:", ix+1);
    scanf("%f", &mas[ix]);
  }

  min = mas[0]; //инициализиране на минималната стойност с първия елемент на масива
  minIx = 0;
  for (ix = 1; ix < n; ix++)
  {
    if (mas[ix] < min) //намерена е нова минимална стойност
    {
      min = mas[ix]; //нова минимална стойност
      minIx = ix;  //индекс на минималния елемент
    }
  }

  printf("Minimalnata stoinost e %f s index %d", min, minIx);

  return 0;
}
```

Прехвърляне на елементи от един масив в друг

```
#include <stdio.h>

#define N 100

int main() {

		int mas1[N], mas2[N], min;
		int n, n_otr, ix, ix_otr, minIx;

		do
		{
				printf("Kolko stoinosti ste vavedete:");
				scanf("%d", &n);
		} while (n < 1 || n > 100);

		for (ix = 0; ix < n; ix++)
		{
				printf("Vavedete %d stoinost:", ix + 1);
				scanf("%d", &mas1[ix]);
		}

		n_otr = 0;
		for (ix = 0, ix_otr = 0; ix < n; ix++)
		{
				if (mas1[ix] % 2 == 0)
				{
						mas2[ix_otr++] = mas1[ix];
						n_otr++;
				}
		}
//след цикъла n_otr = ix_otr, така че променливата n_otr може да отпадне
		printf("Parvi masiv:\n");
		for (ix = 0; ix < n; ix++)
				printf("Element[%d] = %d\n", ix, mas1[ix]);

		printf("Vtori masiv:\n");
		for (ix = 0; ix < n_otr; ix++)
				printf("Element[%d] = %d\n", ix, mas2[ix]);

		return 0;
}
```

Символни низове

Символен низ

- Масив от символи
- терминираща 0 – символ ‘\0’ с asci код 0 – указва края на символния низ
- char text[10]; - позволява съхраняването на 9 символа + терминираща 0
- Форматиращ спецификатор %s
  - printf(“%s”, text);
  - scanf(“%s”, text);
- Функции за вход/изход на низове
  - puts
  - gets

Специализирани функции в заглавен файл <string.h>

- **char *strcat(char *dest, const char *src)** - Appends the string pointed to, by src to the end of the string pointed to by dest.
- **int strcmp(const char *str1, const char *str2)** - Compares the string pointed to, by str1 to the string pointed to by str2.
- **char *strcpy(char *dest, const char *src)** - Copies the string pointed to, by src to dest.
- **size_t strlen(const char *str)** - Computes the length of the string str up to but not including the terminating null character.
- **char *strstr(const char *haystack, const char *needle)** - Finds the first occurrence of the entire string needle (not including the terminating null character) which appears in the string haystack.

Сравнение на низ с предварително зададен

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

int main()
{
	int chisla[] = { 5, 8, 3, -3 };
	char predef_pass[] = "parola";
	char user_pass[20]; //може да съхрани до 19 символа + терминираща 0
	int tries = 0;

	do {
		printf("Enter pass:");
		scanf("%19s", user_pass); //ограничава входа до 19 символа
		tries++;
		if (tries >= 3 && strcmp(predef_pass, user_pass))
			return 5;
	} while (strcmp(predef_pass, user_pass));

	printf("Welcome.");

	return 0;
}
```