**Л Е К Ц И Я 6**

**Масиви C**

Всяка ***променлива,***разгледана до тук има едно име, една стойност (която може да се промени), един тип и адрес. Когато се налага да се обработят множество данни е удобно да се използва структурата от данни ***масив***, която се асоциира с едно име, един начален адрес, един тип и множество от стойности.

**1. Понятие за масив**

• Подредена последователност от еднотипни стойности с общо ***име***, които наричаме ***елементи***на масив.

•  Елементите са разположени плътно в последователни полета на паметта.

•  Всеки елемент на масива има пореден номер, наречен индекс (цяло число).

Друга дефиниция на масив: статична подредена колекция от елементи от еднакъв тип с равнопоставен достъп чрез изчисляем(и) индекс(и) до всеки елемент .

**2. Декларация на масив**

< тип > <име на масив > [ брой елементи ];

•  Тип: всички типове на С

•  Име: се съставя по правилата за съставяне на идентификатори

• Брой елементи: израз от цял тип, който може да има само положителни стойности и не може да бъде променлива, определя броя на елементите на масива

При деклариране на масива компилатора заделя необходимата памет за него, започвайки от един ***начален адрес***. Името на масива играе роля на указател към този начален адрес. Достъпът до елементите на масива се осъществява посредством индекс (или адрес), като всички елементи имат едно и също име (това на масива), но различни индекси.Пример:

|  |  |
| --- | --- |
| **Int** A [10]; | едномерен масив |
| **Int** A [10] [10]; | двумерен масив |
| **Int**Const [26] |  |
| **Char** name [12] |  |
| **Double** rate [365] |  |

3. Обръщение към компонента от тип масив

T [ i ] i – тата компонента на масива

Индексите на един масив с **n**елемента, започват от **0**и завършват с **n -1**

Достъпът до кой да е елемент на масива се осъществява посредством неговия**индекс**(или адрес)

**4. Инициализация на масив**

|  |  |
| --- | --- |
| int X [4] = {2,5,8,-4} |  |
| float Y [] = {3.05, -4.5, 3,4,5.50} | броят на елементите не е зададен явно |
| char s [] = “tehnikum” |  |
| char c [4] = “ABC” |  |

**5. Видове масиви**

**едномерни**

< тип > <име на масив > [ брой елементи ];

**int** a[5]; - масив с 5 елемента

**двумерни**

< тип > <име на масив > [ брой елементи ] [ брой елементи ];

Първият атрибут за размерност дефинира брой редове, а вторият - брой колони.

**int** a [5][3]; - масив с 5 реда и 3 стълба

**многомерни**

< тип > <име на масив > [ брой елементи ] ……… [ брой елементи ];

**6. Символни масиви**

Символните масиви са едномерни. Броят на елементите се определя от броя на символите (букви и цифри), които формират даден низ. Затова и масивите от този тип позволяват да се обявяват без предварително посочване дължината на масива. Последният елемент се използва за служебни цели в него се зарежда символната константа '\0' за край на низ.

**Типови задачи с числови масиви**

След деклариране на един масив всеки негов елемент може да бъде избран и обработен посредством името на масива и индекса на елемента, заграден в квадратни скоби. Например записа А[2] M [2][3] посочват третия елемент на едномерния масив А и елемента от третия ред и четвъртия стълб на двумерния масив М.

Индексираните елементи на масива се наричат ***индексирани променливи***и могат да се използват в програмата както простите (неиндексирани) променливи. С индексираните променливи могат да бъдат извършвани всички операции, допустими за простите променливи от същия тип.

**А. Въвеждане стойности за елементите на масива и извеждане стойностите на елементите на масива**

В езика C/С++ са предвидени средства за въвеждане и извеждане на цял масив, затова въвеждането и извеждането се извършва *поелементно*. За тази цел се организира цикъл и най – често се използва оператор за цикъл FOR.

Пример:

|  |
| --- |
| /\*въвеждане стойности за елементи на масив\*/ |
| **for** (i=0;i<=11;i++) |
| { |
| cout<<"a["<<i<<"]="; |
| cin>>a[i]; |
| } |
| /\*извеждане стойности на елементи на масив\*/ |
| **for** (i=0;i<=11;i++) |
| cout<<"a["<<i<<"]=”<<a[i]<<"\n"; |

**Б. Намиране max, min елемент на масива**

Задачата за намиране най – големия (най- малкия) по стойност елемент от елементите на масива се свежда до следната идея: Последователно, от първия към последния елемент на масива, се обхождат всички елементи на масива; като най-голямата стойност се помни в междинна променлива (например в MAX); всеки преглеждан елемент се сравнява с МАХ и ако е по – голям, стойността на МАХ се променя и става равна на стойността на този елемент. По този начин на всяка стъпка МАХ ще съдържа най – голямата стойност, открита до тази стъпка. След като се обходят всички елементи на масива, МАХ ще има стойността на на – големия елемент.

За начална стойност на МАХ се избира първия елемент на масива. Този алгоритъм може да се приложи и за намиране на най – малкия елемент на масива.

Пример:

|  |
| --- |
| /\*намиране max/min елемент\*/ |
| **int** max=a[0],min=a[0]; |
| **for** (i=1;i<=11;i++) |
| { |
| **if** (max<a[i]) |
| max=a[i]; /\* намиране на max елемент \*/ |
| **if** (min>a[i]) |
| min=a[i]; /\* намиране на min елемент \*/ |
| } |
| cout<< "макс.елемент е " <<max<<"\n"; |
| cout<< "миним.елемент е "<<min<<"\n"; |

**В. Сумиране елементите на масив без условие/при условие**

Въвежда се променлива (с име SUMA, SUMA1) в която ще се натрупва сумата от стойностите на елементите на масива:

|  |
| --- |
| /\*сума на всички елементи на масива \*/ |
| **int** suma=0,suma1=0; |
| **for** (i=0;i<=11;i++) cin>>a[i]; |
| suma=suma+a[i]; /\*сума без условие\*/ |
| cout<<"сума на всички елементи на масив ="<<suma<<"\n"; |
| **for** (i=0;i<=11;i++) |
| if (a[i]>0) |
| suma1=suma1+a[i]; /\*сума с условие – сума на всички положителни елементи \*/ |
| cout<<"сума на положителните елементи на масив="<<suma1<<"\n"; |

**Г. Броене елементи на масив с условие**

Алгоритъмът, по кoйто се решават задачите за броене има следните стъпки:

•  Предварително на променлива от цял тип (например br) се присвоява стойност 0 (в края br ще съдържа броя на елементите, отговарящи на поставеното условие)

•  Последователно се обхождат всички елементи, като за всеки от тях се проверява дали отговаря на условието; ако елементът отговаря на условието, стойността на променливата се увеличава с 1.

|  |
| --- |
| /\*броене на елементите на масив \*/ |
| **int** br=0; |
| **for** (i=0;i<=11;i++) cin>>a[i]; |
| **if** (a[i]>=0) |
| br=br+1; /\*брояча се увеличава с 1\*/ |
| cout<<"брой полож.елементи в масива ="<<br<<"\n"; |

**Д. Търсене на елемент по определено условие**

Може да се търси ученик с най – висок успех, фирма с най – голям оборот и др. Задачата за търсене се свежда до това да се зададе условие, наречено **критерий за търсене**. Последователно, от първия към последния елемент на масива, се обхождат всички елементи на масива; всеки елемент се проверява дали отговаря на поставеното условие за търсене; като резултат елементът може да бъде намерен или не.

*Тази задача прилича на задачата за броене елементи на масив при определено условие*.

|  |
| --- |
| /\*търсене на елементите на масив по условие\*/ |
| **int** br=0; |
| **for**(i=0;i<=11;i++) cin >>a[i]; |
| **if**(a[i]=5 ) /\* условие за търсене \*/ |
| br=br+1; /\*брояча се увеличава с 1\*/ |
| cout<<"броя на елементите = "<<br<<"\n"; |

**Е. Размяна на стойностите на елементите на масив**

Тази операция се извършва с помощта на допълнителна променлива от типа на масива.

Като пример ще използвам размяната на стойностите на първия и третия елемент на масива А (Р е променлива от от типа на масива):

|  |
| --- |
| p=a[1]; |
| a[1]=a[3]; |
| a[3]=p; |

Масивите са градивен елемент на езика С за съхраняване на елементи от един и същи тип. Mасивите са по-неудобни за работа - например техният размер се задава при дефинирането и не може да се променя.

Задаване на стойности на елементите на масив.   
**const int SAL\_MAXSIZE = 10;**   
**double sal[SAL\_MAXSIZE]; /\* фиксираме дължината на масива \*/**   
**int sal\_size = 0;        /\* брой на използваните елементи на масива \*/**   
**bool more = true;**   
**/\* необходими са 2 условия за прекратяване на цикъла - спиране на въвеждането или запълване на всички елементи на масива \*/**   
**while (more and sal\_size < SAL\_MAXSIZE)**   
**{ cout << "Enter salary or 0 to quit";**   
**double x;**   
**cin >> x;**   
**if (cin.fail()) more = false**   
**else**   
**{ sal[sal\_size] = x; /\* задаване на стойност на елемент на масива \*/**   
**sal\_size++;        /\* увеличаване с 1 на използваните елементи \*/**   
**}**   
**}**

**Mасивите като параметри на функции**  
    За да се обозначи, че един формален параметър на функция е масив, след името му се пише []. Масивите винаги се предават чрез псевдоним (по адрес). За масив - входен параметър на функция обикновено се използва константен псевдоним.   
**double maximum(const double a[], int a\_size)**   
**{  if (a\_size == 0) return 0;**   
**double highest = a[0];**   
**int i;**   
**for (i = 1; i < a\_size; i++)**   
**if (a[i] > highest) highest = a[i];**   
**return highest;**   
**}**   
    Следващата програма чете заплати от стандартен вход и отпечатва максималната заплата:   
**// salarray.cpp**   
**#include <iostream>**   
**using namespace std;**

**void read\_data(double a[], int a\_maxsize, int& a\_size)**   
**{ a\_size = 0;**   
**double x;**   
**while (a\_size < a\_maxsize and (cin >> x))**   
**{ a[a\_size] = x; a\_size++; }**   
**}**

**double maximum(const double a[], int a\_size)**   
**{ if (a\_size == 0) return 0;**   
**double highest = a[0];**   
**int i;**   
**for (i = 1; i < a\_size; i++)**   
**if (a[i] > highest) highest = a[i];**   
**return highest;**   
**}**

**int main()**   
**{ const int SALARIES\_MAXSIZE = 100;**   
**double salaries[SALARIES\_MAXSIZE];**   
**int salaries\_size = 0;**

**cout << "Please enter all salary data: ";**   
**read\_data(salaries, SALARIES\_MAXSIZE, salaries\_size);**

**if (salaries\_size == SALARIES\_MAXSIZE and not cin.fail())**   
**cout << "Sorry - extra data ignored\n";**

**double maxsal = maximum(salaries, salaries\_size);**   
**cout << "The maximum salary is " << maxsal << "\n";**   
**return 0;**   
**}**

|  |
| --- |
| **Please enter all salary data: 200 240 250 210 290 310 205 q**  **The maximum salary is 310** |

**Масиви от символи.**    Съхраняването и обработката на низове в езика С (т.н. С-низове ) се извършва в масиви от символи. Всеки низ завършва със специален символ с ASCII код 0.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **char greeting[6] = "Hello";**  **char greeting[] = "Hello";**  **char greeting[10] = "Hello";** | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Символ | **H** | **e** | **l** | **l** | **o** | **'\0'** | | Индекс | 0 | 1 | 2 | 3 | 4 | 5 | |

Превръщането С-низ в типа **string** и обратно става по следния начин:

|  |  |
| --- | --- |
| **string s = "ABCD";**  **char c[] = s.str();** | **char c[] = "1234";**  **string s = static\_cast<string>(c);** |

Пример с използване на функцията **atoi** от библиотеката **cstdlib** за превръщане на низ в число:   
**string year = "1999";**   
**int y = atoi(year.c\_str());**   
    Важно е да не се забравя символа за край на низ при използване на масиви от символи, защото много функции, които работят със С-низове и операцията изходен поток определят дължината на низа по този символ.   
    Използването на български букви в типа **string** създава някои проблеми. Обикновено може да се работи нормално с член-функциите на класа, но не може да се извличат букви от низа, защото елементите на **string** са от тип **char**, който съхранява числа от -127 до 127. Решението е да се записват низове с български букви като С-низове с елементи от тип **unsigned char** (числа от 0 до 255) по следния начин:   
**unsigned char s\_bul[] = "АБВГ";**

**Двумерни масиви.** За съхранение и обработка на матрици се използват двумерни масиви. При дефиниране на масива се задават броя на редовете и броя на стълбовете.   
**// matrix.cpp**   
**#include <iostream>**   
**#include <iomanip>**   
**#include <cmath>**   
**using namespace std;**

**const int BAL\_ROWS = 6;**   
**const int BAL\_COLS = 5;**

**double future\_value(double init\_bal, double p, int nyear)**   
**{  double b = init\_bal \* pow(1 + p/12/100, 12\*nyear);**   
**return b;**   
**}**

**/\* втората размерност на двумерния масив се задава като константа \*/**   
**void print\_table(const double table[][BAL\_COLS], int table\_rows)**   
**{  int i, j;**   
**cout << fixed << setprecision(2);**   
**for (i = 0; i < table\_rows; i++)**   
**{  for (j = 0; j < BAL\_COLS; j++)**   
**cout << setw(10) << table[i][j];**   
**cout << "\n";**   
**}**   
**}**

**int main()**   
**{  double bal[BAL\_ROWS][BAL\_COLS]; /\* дефиниране на двумерен масив \*/**   
**/\* двоен цикъл за обхождане на елементите на двумерен масив \*/**   
**for (int i = 0; i < BAL\_ROWS; i++)**   
**for (int j = 0; j < BAL\_COLS; j++)**   
**bal[i][j] = future\_value(10000, 5+i\*0.5, 5+j\*5);**   
**print\_table(bal, BAL\_ROWS);**   
**return 0;**   
**}**

|  |
| --- |
| **12833.59  16470.09  21137.04  27126.40  34812.90**  **13157.04  17310.76  22775.84  29966.26  39426.72**  **13488.50  18193.97  24540.94  33102.04  44649.70**  **13828.17  19121.84  26442.01  36564.47  50561.98**  **14176.25  20096.61  28489.47  40387.39  57254.18**  **14532.94  21120.65  30694.52  44608.17  64828.80** |

Ето още един пример:

#include <stdio.h>

/\* count digits, white space, others \*/

main()

{

int c, i, nwhite, nother;

int ndigit[10];

nwhite = nother = 0;

for (i = 0; i < 10; ++i)

ndigit[i] = 0;

while ((c = getchar()) != EOF)

if (c >= '0' && c <= '9')

++ndigit[c-'0'];

else if (c == ' ' || c == '\n' || c == '\t')

++nwhite;

else

++nother;

printf("digits =");

for (i = 0; i < 10; ++i)

printf(" %d", ndigit[i]);

printf(", white space = %d, other = %d\n",

nwhite, nother);

}

На изхода на тази програма ще получим:

**digits = 9 3 0 0 0 0 0 0 0 1, white space = 123, other = 345**

ПРИМЕР

int number[20], \*p=number;

1) запазва място за 20 последователни елемента –int числа

2) numberпредставлява константен указател към първия

елемент