ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Урок №6

Програмиране на език C

Съдържание

1. Необходимост от групиране на данни

Днес ще говорим с вас за запазване на данни. На един от първите уроци ние се запознахме със съществуването на променливите и ги определихме като отрязък на оперативната памет за разместване на информация. Несъмнено е че нормалната програма не може да съществува без променливи, обаче няколко обикновени променливи не решават проблема с оперирането на данни. А работата е в това, че всяка от променливите разгледани в предишните уроци е способна да съхранява едновременно само един елемент от информацията. За да се съхрани втори е необходимо да се създаде още една променлива. Но какво ще правим ако ни се наложи да запазим множество елементи от еднороден тип данни. Ще бъде съвсем неудобно да се създават за всеки елемент променливи. Ами ако ни се наложи да работим със стотици елементи? Задачата бързо се превръща неизпълнима. Да се създават няколко стотин променливи е безумие.

Как тогава да решим такава на пръв непосилна задача?! В нашия случай решението се крие в така наречените масиви. Да разгледаме определението им и техните особености.

Понятието масив.

1. Масив — това е съвкупност от променливи, който позволява да се запазват няколко еднотипни значения.

Всичките значения на тази съвкупност са обединени под едно име

3. При това всяка променлива в масива е самостоятелна единица наречена елемент.
4. Всеки елемент има свой пореден номер – индекс. По индекса може да се обърнем конкретен елемент на масива.
5. Номерацията на елементите в масива започва от нула.

Схема:

Изхождайки от гореописаните утвърждения общата схема за представяне на масива ще изглежда така:

Разполагане на масива в паметта:

Масивът се разполага в паметта последователно, елемент след елемент. Първо е нулевият след това първият и т.н. Елементите се разполагат според нарастването на адреса:
Един елемент на масива изостава от друг на количество байт, равно на базовия тип на масива. Формулата по която се произвежда позициониране по масива:

Базов адрес + размер на базовия тип * индекс;

Ако се посочва неправилен адрес, протича позициониране на базовия адрес на адрес, изчислен по формули. По този начин програмата получава пълен достъп до съдържанието на клетката памет, която по принцип не й принадлежи. В резултат на това може да произлезе грешка на етапа за изпълнение.

За заключение, трябва да се отбележи, че всеки елемент от масива има своя собствена размерност, която пряко зависи от типа на целия масив. Например, ако масива има тип данни int- размера на всеки елемент в него е 4 байта. По такъв начин, общия размер на целия масив се изчислява по формулата:

ОБЩ_РАЗМЕР = РАЗМЕР_НА_ТИПА_ДАННИ*КОЛИЧЕСТВО_ЕЛЕМЕНТИ_В_МАСИВА

Сега на теория ние знаем почти всичко за масивите. Остава само да се запознаем с практическата част и да се убедим колко лесно и удобно се създават и използва дадена конструкция. За това да преминем към следващия раздел на урока.

2. Създаване на масива и запълването му с данни

Синтаксис за обявяване на масива.

За начало ви е необходимо да се научите да създавате масиви. А за това първо да определите общия синтаксис. Второ – да се изясни по какви правила и ограничения ще работи този синтаксис.

```
тип_данни име_на_масива[количество_елементи];
```

1. **тип_на_данните** — всеки от съществуващите известни на вас типове данни. Именно тези типове ще се състоят във всеки елемент от масива.
2. **име_на_масива** — всяко име което се подчинява на правилата за имена на променливите (тези правила сме ги разглеждали с вас на първия урок).
3. **количество_на_елементите** — числото на елементите в масива. На даденото място трябва да се намира – целочислено константно значение. Такова значение може да бъде или целочислен литерал, или константна целочислена променлива.

Забележка: Обърнете внимание, че количеството елементи на масива трябва да бъде определено на етапа на създаването на програмата. С други думи да се зададе размера на масива според зависимостта на някакво условие или по решение на потребителя е невъзможно. Това ще приведе към грешка на етапа за компилация.

Първи вариант.

Обявен е масива ar, съставен от 5 елемента, всеки от които има тип данни int.

```
int ar[5];
```

Вариант втори.

Обявена е константата size, значението на която е равно на 3, а след това масив br, съставен от 3 елемента, всеки от които имат тип данни double.

```
const int size=3;

double br[size];
```

Забележка: Препоръчваме ви да използвате втората форма на запис, защото тя е по-коректна и удобна.

Обръщение към елементите на масива.

Да разгледаме как да се обърнем към конкретен елемент от масива.

```
Запис на значение име на
масива[индекс_на_елемента]=значение;
получаване на значение
cout<<имя_масива[индекс_элемента];
```

Тук на мястото на индекс_на_елемента може да се сложи всяко целочислено значение, в това число и изрази чийто резултат е цяло число.

```
const int size=5;
int ar[size]; // създаване на масив
ar[2]=25; // запис на значението 25 в елеме
с индекс 2 // извеждане на значението на
екран елемент с индекс 2 - 25
cout<<ar[2]<<"\n";
```

Забележка: Още един път напомняме – номерацията на елементите в масива започва от нула! По такъв начин в масив от 5 елемента последния елемент има индекс 4. Не трябва да се излиза извън пределите на масива, това ще доведе до грешка на етапа за изпълнение.

Варианти за инициализация на масива:

Запълването на масива с данни става по два начина:

Първи начин — инициализация при създаването.

тип_на_данните име_на_масива[количество елементи]={значение1, значение2, … значение n};

```
const int size=3;
int ar[size]={1,30,2};
```

При такава форма за инициализация има няколко особености:

1. Всички значения от списъка инициализации имат също такъв тип данни както и масива, затова при създаването количеството елементи може и да не се уточнява. Операционната система сама ще определи размера на масива изхождайки от броя на елементите в списъка инициализация.

Тип_на_данни имя_на_маива[]={значение1, значение2, значение3, … значение n};

```
int ar[]={1,30,2};  /*В дадения ред масива автоматично ще получи размер 3.*/
```

2. Ако броя на елементите в списъка за инициализация е по малко от броя елементи в масива, то останалите значения автоматично се попълват с нула:

```
int ar[5]={1,2,3}
```

такъв запис е еквивалентен на записа:

```
int ar[5]={1,2,3,0,0};
```

3. Ако значенията в списъка за инициализация са повече от количеството елементи в масива, тогава произлиза грешка в етапа за компилация:

```
int array[2]={1,2,3}; // грешка при етапа за компилация
```

4. При инициализация на масива може да се използва вече известната ви унифицирана инициализация

```
int arr[]{1, 2, 3};
int arr2[4]{11, 21, 31};
```

Втори начин — инициализация на масива с помощта на цикъл.

В такъв случай запълването на масива със значения може да стане с помощта на потребителя. Име на проекта InitArray.

```
#include<iostream
> using namespace
std; void main()
{
   const int size=3;
    int ar[size]; //създаване ма масив от три
    елемента
    //цикл подбиращ елементи от масива
   for (int i=0;i<size;i++)
    {
       cout<<"Enter element\n";
   /* на всяка итерация от цикъла потребителя поставя елемент с индекс i за
запълване. Тайната е в това, че i е всеки път с ново значение */
       cin>>ar[i];
    }
}
```

Изкарване на съдържанието на масива на екрана.

Вече сигурно сте се досетили, че повечето операции с масиви е по удобно да се повеждат с помощта на цикъл, избирайки елементите по ред. Това действително е така и изкарването на екрана не е изключение. Да изкараме пример на пълна програма, която създава запълва и показва на екрана масива. Име на проекта ShowArray.

```
#include<iostream
> using namespace
std; void main()
{
   const int size=3;
    int ar[size]; //създаване на масива от три
    елемента//цикъл подбиращ елементи от масива
   for (int i=0;i<size;i++)
    {
       cout<<"Enter element\n";
    /* на всяка итерация от цикъла потребителя поставя елемент с индекс i за
запълване. Тайната е в това, че i е всеки път с ново значение */
       cin>>ar[i];
    }
    cout<<"\n\n";
    //цикъл подбиращ елементи от
   масива for (int i=0;i<size;i++)
    {
       //изкарване на елемента с индекс
       i на екрана cout<<ar[i]<<"\n";
    }
}
```

Сега , когато с вас сме се запознали с масивите, нека ад преминем към следващия раздел на урока и да разгледаме няколко практически примера за работа с тях.

3. Пример за програма за намиране на сумата отрицателни елементи от масива

Постановка на задачата:

Да се напише програма, която да намира сумата на всички отрицателни значения в масива. Име на проекта AmountOfNegative.

Код за реализация:

```
#include <iostream>
using     namespace
std; void main ()
{
         //определяне размера на
       масива const int size=5;
         //създаване и инициализация на масива с
       данни int ar[size]={23,-11,9,-18,-25};
         //променлива за насъбиране
       на сума int sum=0;
         //цикъл подбиращ по ред елементи от
       масива for (int i=0;i<size;i++)
         {
            //ако значението на елемента е отрицателно
            (по-малко от нула) if(ar[i]<0)
           sum+=ar[i]; //да се добави неговото значение кум общата
           сума
         }

         //показване значението на сумата
       на екранаcout<<"Sum =
       "<<sum<<"\n\n";
}
```

Коментар към кода:

1. Цикъла подред избира елементи от 0 до size. При това size не влиза в проверяемия диапазон, тъй като индекса на последния елемент е size-1.

2. На всяка итерация на цикъла протича проверка на съдържимото в елемента за отрицателно значение

3. Ако значението е по малко от нула, то се прибавя към сумата.

Както виждате работата с масивите на много прилича на анализ на някакъв диапазон – 0, а максимално – се определя количеството на елементи в масива.

Пример за програма намиращата минималните и максималните елементи от масива.

Постановка на задачата:

Да се напише програма, която да намира минималното и максималното значение в масива и да ги показва на екрана. Име на проекта е MinMaxElement.

Код за реализация:

```
#include <iostream>
using     namespace
std; void main ()
{
         // Определяне  количествата  елементи
       на масива const int size=5;
         // Създаване и инициализация на масива
       int ar[size]={23,11,9,18,25};

       int max=ar[0]; // нека 0 е максимален елемент

       int min=ar[0]; // нека 0 е минимален елемент

         //цикъла подбира елементи от масива
       започвайки с 1-ы for (int i=1;i<size;i++)
         {
            //ако текущия елемент е по-малък от минимума
```

```
            if(min>ar[i])
              //да се презапише
            значението на минимума
            min=ar[i];
            //ако текущия елемент е по-голям от
            максимума if(max<ar[i])
              //да се презапише
            значението на максимума
            max=ar[i];
          }

          // изкарване резултата на
        ектан cout<<"Max =
        "<<max<<"\n\n"; cout<<"Min =
        "<<min<<"\n\n";
}
```

Коментар към кода:

1. За начало, изкарваме предположение, че минималния елемент на масива е с индекс 0.
2. Записваме значението на елемента с индекс 0 в променливата min.
3. След това, за да потвърдим или опровергаем този факт, преглеждаме всички елементи от масива започвайки с елемента с индекс 1 в цикъла.
4. На всяка итерация от цикъла, сравняваме предполагаемия минимум с текущия елемент от масива (елемент с индекс i).
5. Ако се срещне значение по-малко, от предполагаемия минимум – значението min се презаписва на най-малкото намерено значение и анализа продължава..

Всички гореописани действия са еднакви и за максимума, само че е необходимо а се търси най-голямото значение. Сега, когато сте запознати с масивите и сте разгледали няколко примера е време да направите нещо сами. Желаем ви успех в преминаването на теста и домашната работа.

4. Домашна работа

За входни данни във всичките описани долу задачи е масив от 10 елемента, запълнен от потребителя с клавиатура.

1. Да се напише програма, която да изкарва съдържанието на масива на обратно.

```
Пример: масив 23 11 6 се превръща в 6 23 11.
```

2. Да се напише програма, която да намира сумата на четните и нечетните елементи от масива.
3. Да се напише програма, която да намира в масива значение, повтарящи се два или повече пъти и да ги покаже на екрана.
4. Да се напише програма, която да намира в масива най-малкото нечетно число и да го покаже на екрана.