Работа с одномерными массивами
Файлы и работа с ними в C/C++

Чернецов Андрей Михайлович,

К.т.н. доцент каф. ПМИИ

Список литературы по курсу

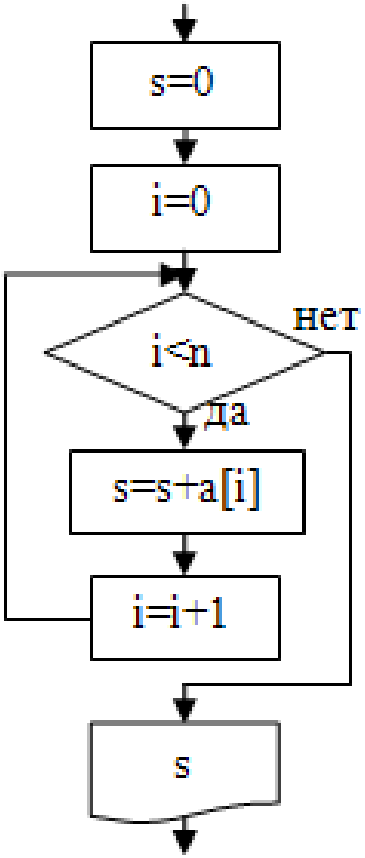
- 1. Князев А.В. Основы языка С++. Учебное пособие. М.: Издательство МЭИ, 2013 – 80 с. ISBN 978-5-7046-1425-8.
- 2. Князев А.В. Работа со сложными структурами данных на языке С++. Учебное пособие. М.: Издательство МЭИ, 2015 – 48 с. ISBN 978-5-7046-1658-0
- 3. Программирование. Сборник задач. Учебное пособие. Санкт-Петербург: Лань, 2019 – 140 с. ISBN 978-5-8114-3857-0

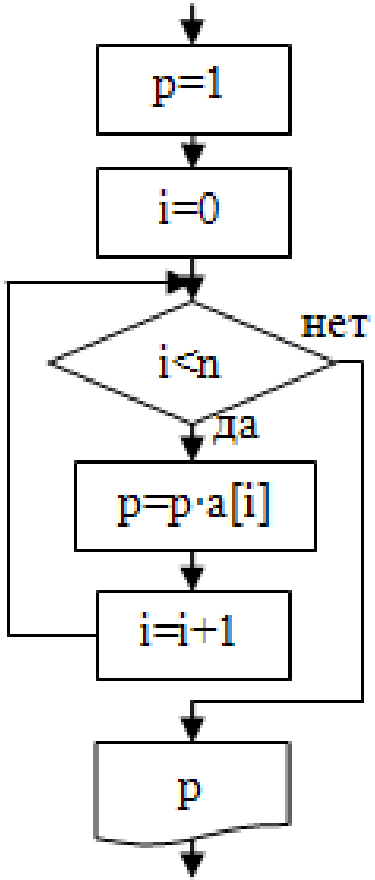
URL: https://e.lanbook.com/book/121485

ТИПОВЫЕ АЛГОРИТМЫ

ОДНОМЕРНЫЕ МАССИВЫ

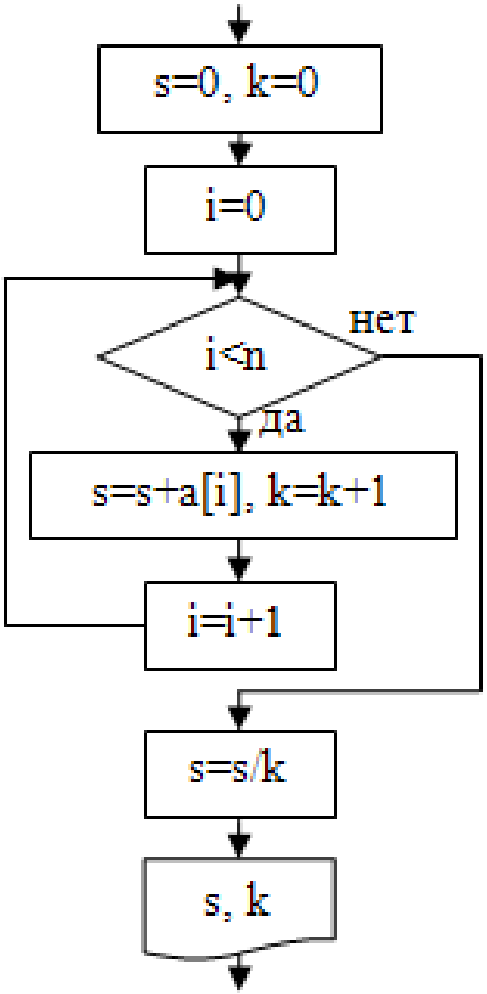
Найти сумму (произведение) элементов массива






```
s=0;
 for(i=0; i<n; i=i+1){
   s=s+a[i]; }

cout<<"s="<<s<<endl;

p=1;
 for(i=0; i<n; i=i+1){
   p=p*a[i]; }
cout<<«P="<<p<<endl;
```

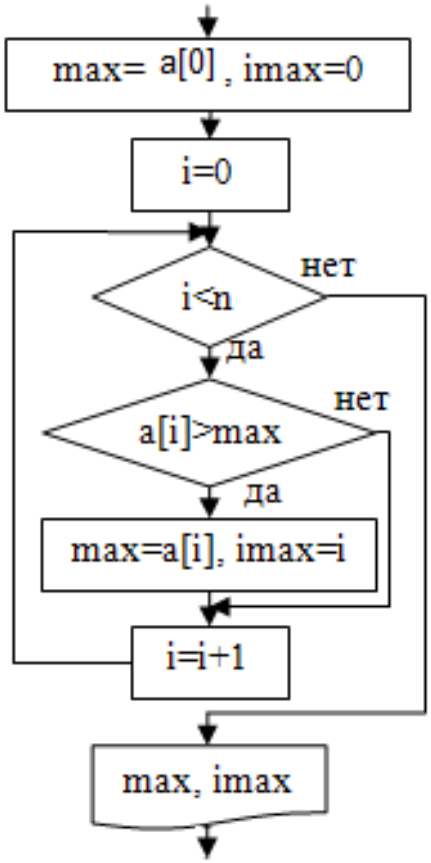
Нахождение среднего арифметического и количества элементов




```
s=0; k=0;
 for(i=0; i<n; i=i+1){
    s=s+a[i];
    k=k+1;
   }

   s=s/k;

 cout<<"s="<<s<<endl;

  cout<<"k="<<k<<endl;
```

K=0 ?

Нахождение максимального элемента массива и его номера



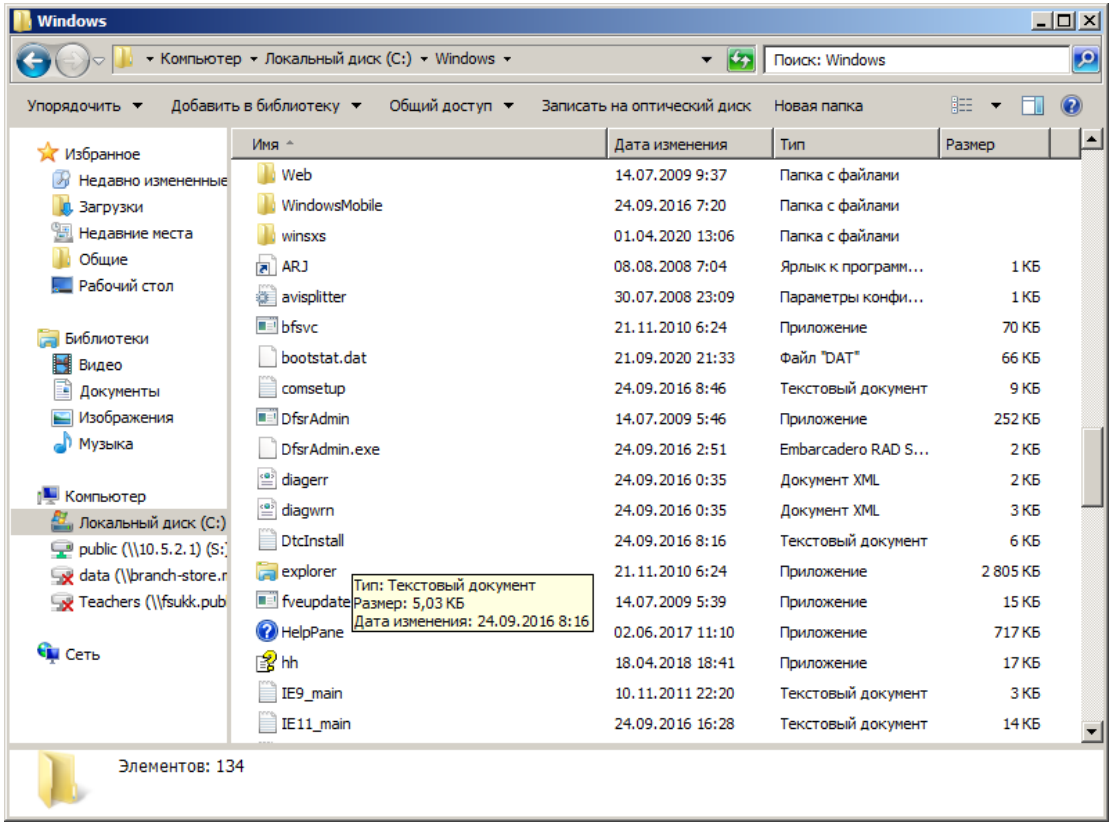
~~max=0~~

```
max=a[0];
imax=0;
for(i=0; i<n; i=i+1) {
   if(a[i]>max){
      max=a[i]; imax=i;
   }
 }
cout<<"max="<<max<<" imax="<<imax<<endl;
```

Файлы и работа с ними

Понятие файла

Файл— именованная область данных на носителе информации.



Очень часто бывают задачи, в которых необходимо производить запись в файл или чтение из файла.

1)Если программа производит много вычислений, по результатам которых нужно построить график эксперимента, например в пакете в MS Excel.

2) Другой пример: программа работает с большим набором данных, которые хранятся в файлах.

Файлы в языке С

- Для программиста открытый файл представляется как последовательность считываемых или записываемых данных.
- При открытии файла с ним связывается поток ввода-вывода. Выводимая информация записывается в поток, вводимая информация считывается из потока.

- Существует 3 стандартных потока:
-поток ввода (клавиатура)
-поток вывода (экран)
-поток ошибок (обычно экран)

- Когда поток открывается для ввода-вывода, он связывается со стандартной структурой типа FILE, которая определена в stdio.h. Структура FILE содержит необходимую информацию о файле.

Файлы в Си

Открытие файла осуществляется с помощью функции fopen(), которая возвращает ссылку на структуру типа FILE, которую можно использовать для последующих операций с файлом.

FILE *fopen(name, type);

name – имя открываемого файла (включая путь),
type — указатель на строку символов, определяющих способ доступа к файлу:
"r" — открыть файл для чтения (файл должен существовать);
"w" — открыть пустой файл для записи; если файл существует, то его содержимое теряется;
"a" — открыть файл для записи в конец (для добавления); файл создается, если он не существует;
"r+" — открыть файл для чтения и записи (файл должен существовать);
"w+" — открыть пустой файл для чтения и записи; если файл существует, то его содержимое теряется;
"a+" — открыть файл для чтения и дополнения, если файл не существует, то он создаётся.

Файлы в Си

Возвращаемое значение — ссылка на открытый поток. Если обнаружена ошибка, то возвращается значение NULL.

fclose() закрывает поток или потоки, связанные с открытыми при помощи **fopen**() файлами. Закрываемый поток определяется аргументом fclose().

Возвращаемое значение: значение 0, если поток успешно закрыт; константа EOF, если произошла ошибка.

Пример на Си

```
#include <stdio.h>
#include <conio.h>
int main() {
  FILE *fp;

  char name[] = "my.txt";

  if ((fp = fopen(name, "r")) == NULL)
  {
    printf("Не удалось открыть файл");
    getch();
    return 0;
  }
  // открыть файл удалось
  ...       // требуемые действия над
данными
  fclose(fp);
  getch();
  return 0;
}
```

Ввод-вывод с файлами на С

Функции fscanf() и fprintf() аналогичны функциям scanf() и printf(), но работают с файлами данных, и имеют первый аргумент — имя файловой переменной.

fscanf(поток, "ФорматВвода", аргументы);

fprintf(поток, "ФорматВывода", аргументы);

Файлы на Си – пример ввода

```
#include <stdio.h>

int main()
{
  FILE *S1, *S2;
  int x, y;

  printf("Введите число : ");
  scanf("%d", &x);
  S1 = fopen("S1.txt", "w");
  fprintf(S1, "%d", x);
  fclose(S1);
  S1 = fopen("S1.txt", "r");
  S2 = fopen("S2.txt", "w");
  fscanf(S1, "%d", &y);
  y += 3;
  fclose(S1);
  fprintf(S2, "%d\n", y);
  fclose(S2);
return 0;
}
```

C++: Файловые потоки

Чтобы программа могла взаимодействовать с файлом, необходимо использовать переменную специального типа - файловый поток.

Такая переменная задается ключевым словом **fstream**.

Для работы с файловым потоком необходимо:

1. подключить библиотеку fstream:

```
#include <fstream>
```

1. объявить переменную типа файловый поток:

```
fstream f;
```

3. открыть файл:
    1. для записи в файл: f.open("1.txt", ios::out);
    2. для чтения из файла: f.open("1.txt", ios::in);
4. произвести запись в файл или чтение из файла:
    1. для записи в файл: f<<"x="<<x;
    2. для чтения из файла: f>>x;
5. закрыть файл: f.close();

!При открытии файла на запись файл создается в папке с выполняемым файлом!

- Если файл уже существует, то все его содержимое содержимое стирается.

- Если требуется выводить информацию в конец уже существующего файла,

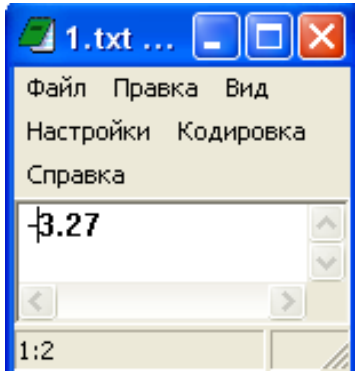
то при открытии файла надо использовать строку:

f.open("1.txt", **ios::app**);

вместо **ios::out**

Пример работы с файлом

1. Создадим файл 1.txt и запишем туда число



2. Файл находится в одной папке с выполняемым файлом!

Пример работы с файлом

Ввод из файла

```
#include "stdafx.h"
#include <iostream>
 #include <fstream>
 using namespace std;
int main()
{
   double x;
   fstream f;
   f.open("1.txt", ios::in);
   f>>x;
   f.close();
   cout<<"x="<<x<<endl;
   return 0;
}
```

Пример работы с файлом

Вывод в файл

```
#include "stdafx.h"
#include <iostream>
 #include <fstream>
 using namespace std;

 int main() {

double a, b;
fstream ftxt, fxls;

 a=5.0; b=-20.11;
 cout<<"a="<<a<<" b="<<b<<endl;
ftxt.open("1.txt", ios::out);
fxls.open("2.xls", ios::out);
ftxt<<"a="<<a<<" b="<<b;
fxls<<"a=\t"<<a<<"\tb=\t"<<b;
ftxt.close();
fxls.close();
 return 0; }
```

Аргументы командной строки

Аргументы командной строки

Используются для указания программе каких-то особенностей работы

Например, широко известен ключ команды /help – вывести справочную информацию по команде.

Также в качестве параметров часто выступают имена файлов – входного и выходного

Функция main

Было

```
int main ()
{

..
return 0;
}
```

Стало

```
int main (int argc, char *argv[])
{

…
return 0;
}
```

argc - число параметров командной строки
argv[] – массив параметров.

Пример работы с параметрами командной строки

```
include <stdio.h>

 int main(int argc, char *argv[])
 {
 int i;
  printf("%d\n", argc);
   for (i=0; i < argc; i++)
      puts(argv[i]);

return 0;
}
```

puts – стандартная функция из stdio.h
Выводит свой аргумент на экран с новой строки