Массивы в С

Массив — это такая структура данных языка С, в которой:

1. хранятся однотипные элементы,
2. располагаясь в памяти последовательно друг за другом
3. в порядке возрастания индексов

Доступ к элементам массива

Осуществляется по индексу:

A[i] – i-й элемент массива.

Замечания:

1. поиск по индексу происходит очень быстро;
2. индекс начинается от 0 (нуля!);
3. индекс элемента — это НЕ НОМЕР элемента!
4. индекс N-го элемента — N-1;
5. выход за границы массива никем не контролируется;

Адрес начала массива — это адрес первого элемента.

Внимание!

В языке С обращение к элементу массива — это *прямое обращение к области памяти* по адресу элемента, который *вычисляется по его индексу*, и интерпретируется в соответствии с типом массива.

Отсюда и опасность бесконтрольного выхода за границы массива, отсюда и интересная возможность — посмотреть, а что там в памяти вообще есть?.. Однако, правильно ли мы интерпретируем те данные, до которых сможем дотянуться?

Кстати, обращение с отрицательными индексами не вызывает у компилятора вопросов:

A[-1] — это *предыдущий* элемент, который лежит в памяти за начальной границей, до массива.

Определение и инициализация массива в С

При объявлении массива должна быть выделена область памяти, достаточная для хранения всех его элементов последовательно друг за другом.

Массивы объявляются так:

```
int A[10];
```

При этом будет выделена память в объеме sizeof(int)*10 байт, а `A` — это идентификатор массива, означающий адрес начала выделенной области.

Зарезервированная память для хранения 10 элементов типа int до ее инициализации будет заполнена непредсказуемым мусором. Поэтому элементы массива, как и переменные, нужно перед использованием обязательно инициализировать.

Явно это делается так:

```
for (i = 0; i < N; i++)
    A[i] = i+1;
```

Если элементов массива немного, то можно инициализировать их прямо при объявлении:

```
int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Если инициализировать только часть элементов, то остальные будут инициализированы нулями:

```
int A[10] = {1, 2, 3}; //все остальные =0
```

Допускается не указывать количество элементов, если есть список инициализации:

```
int A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Вопрос

Есть ли ошибка во фрагменте кода:

```
int A[10];
A[10] = 0;
```

Решение

И да, и нет.

С точки зрения компилятора — ошибки нет. (Может программист этого действительно захотел — откуда ему знать?)

А с точки зрения логики — да, выход за границы массива.