4. Массивы

https://github.com/is-itmo-c-22/lectures/blob/main/22.09.19/Lecture 2. Pointer%2C arrays%2C struct and union.pdf

Массив

- Конечное множество однотипных элементов
- Размер множества не меняется
- Индексация с 0
- Многомерные массивы

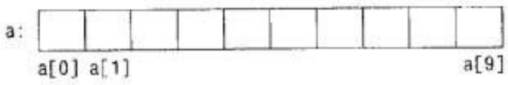
Массив

```c
int main() {
    int arr[10];
    int arr2[] = {1, 2, 3, 4, 5};
    int arr3[3] = {1, 2, 3};
    int arr4[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
    };

    printf("%d\n", arr2[0]);
    printf("%d\n", arr4[1][2]);
}
```

Связь массивов и указателей

- Определим массив
  int a[10];
- Определим указатель
  int *pa;
- Присвоим указатель адресу первого элемента массива
  pa = &a[0];
- Получим значение первого элемента массива через указатель
  int x = *pa;

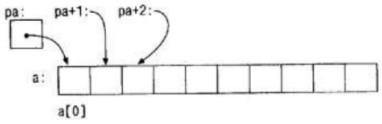


Связь массивов и указателей

- Получим указатель на следующий элемент массива
  *(pa + 1)

- Получим указатель на произвольный элемент массива
  *(pa + i) – это эквивалентно a[i]

- Компилятор преобразует ссылку на массив в указатель на начало массива, следовательно:
  - Имя массива является указательным выражением
  - Записи pa = &a[0] и pa = a эквивалентны
  - Записи a[i], *(a + i), *(pa + i) и pa[i] эквивалентны
  - Массив можно объявлять, как указатель, а потом пользоваться им, как массивом



Указатель на массив — это не указатель на первый элемент (хотя побитово они, конечно, совпадают), здесь нет никакого сведения. Это полноценный тип, который «знает» размер массива. Поэтому при инициализации размеры должны совпадать.

```
int a[4];
int(*pa)[4] = &a; // OK
int(*p2)[2] = &a; // ошибка, размеры не совпадают
```

При инкременте указатель на массив увеличивается на размер всего массива, а не на размер элемента.

6.1. Создание и удаление динамического массива

Если `T` некоторый тип, `n` переменная, значение которой может определяются в процессе выполнения программы, то инструкция

```
T *pa = new T[n];
```

создает массив в динамической памяти. Тип переменной `n` должен приводиться к `std::size_t`, значение может быть нулем. Размер памяти, необходимой для размещения массива, то есть `n*sizeof(T)`, ограничен сверху платформой и компилятором. Переменная `pa` указывает на первый элемент массива.

Если тип `T` тривиальный, то элементы будут иметь случайное значение, в противном случае для инициализации элементов будет использован конструктор по умолчанию.

В C++11 появилась возможность использовать список инициализации.

В C++11 появилась возможность использовать список инициализации.

```
int *pa = new int[n]{1, 2, 3, 4};
```

Если число инициализаторов больше размера массива, то лишние не используются (компилятор может выдать ошибку, если значение `n` известно на стадии компиляции). Если размер массива больше числа инициализаторов, то для оставшихся элементов гарантируется вызов конструктора по умолчанию, в том числе и для тривиальных типов. Таким образам, указав пустой список инициализации, мы гарантируем вызов конструктора по умолчанию для всех элементов массива тривиального типа.

Оператор `new[]` сначала выделяет память для всего массива. Если выделение прошло успешно, то, если `T` нетривиальный тип или есть список инициализации, вызывается конструктор для каждого элемента массива начиная с нулевого. Если какой-нибудь конструктор выбрасывает исключение, то для всех созданных элементов массива вызывается деструктор в порядке, обратном вызову конструктора, затем выделенная память освобождается. Стандартные функции выделения памяти при невозможности удовлетворить запрос выбрасывают исключение типа `std::bad_alloc`.

Динамический массив удаляется оператором `delete[]`, который применяется к указателю, возвращаемому оператором `new[]`.

```
delete[] pa;
```

При этом, если при создании массива использовался конструктор, то для всех элементов массива вызывается деструктор в порядке, обратном вызову конструктора (деструктор не должен выбрасывать исключений), затем выделенная память освобождается.

В остальных отношениях указатель `pa`, возвращаемый оператором `new[]`, является просто указателем на начало массива, через него нельзя (во всяком случае «законно») получить размер массива, этот размер надо хранить отдельно. Соответственно с динамическим массивом нельзя использовать диапазонный `for`. Указатели в C/C++ поддерживают индексатор (встроенный оператор `[]`), поэтому доступ к элементам динамического массива выглядит так же, как и к обычному массиву, контроля за корректностью индекса нет.

https://habr.com/ru/post/495444/