**6.1. Правила описания указателей**

**Описание** указателей:

*int \*a; // указатель на переменную int*

*double \*b; // указатель на double*

*int \*a[5]; // массив из 5 указателей*

*int (\*a)[5]; // указатель на массив*

* Указатели обязательно должны быть инициализированы, особенно с классом памяти *auto*
* Нулевые указатели запрещены

**6,2 Как связаны указатели и массивы?**

*int a[10], b[20], \*p;*

*p = &a[0]; // или p = a*

*// имя массива = адрес его первого элемента*

*p[3] = 0; // эквивалентно a[3] = 0*

*p = &a[2];*

*p[3] = 0; // эквивалентно a[5] = 0*

*p = &b[3];*

*p[3] = 0; // эквивалентно b[6] = 0*

*p--;*

*p[3] = 0; // эквивалентно b[5] = 0*

**Имя массива** – константа, указатель на 0 элемент массива

**Указатель** – переменная

В Си существует связь между указателями и массивами, и связь эта настолько тесная, что эти средства лучше рассматривать вместе. Любой доступ к элементу массива, осуществляемый операцией индексирования, может быть выполнен с помощью указателя. Вариант с указателями в общем случае работает быстрее, но разобраться в нем, особенно непосвященному, довольно трудно.

Между именем массива и указателем, выступающим в роли имени массива, существует одно различие. *Указатель - это переменная*, поэтому можно написать *pa=a* или *pa++*. Но *имя массива не является переменной*, и записи вроде *a=pa* или *a++* не допускаются

Указатели иногда используются для обращения к массивам, поскольку арифметика указателей чаще всего выполняется быстрее, чем индексация массивов.

Преимущество - скорость использования указателей - наиболее заметно, когда осуществляется последовательный доступ к массиву.

**6.3Назначение прототипа функции\**

Прототип указывает имя, тип возвращаемого значения функции и типы ее параметров. это позволяет корректно скомпилировать вызов функции, при этом сама функция может быть написана в другом месте программы

Если используются прототипы, то компилятор может обнаружить любые сомнительные преобразования типов аргументов, необходимые при вызове функции, если тип ее параметров отличается от типов аргументов. При этом будут выданы предупреждения обо всех таких сомнительных преобразованиях. Компилятор также обнаружит различия в количестве аргументов, использованных при вызове функции, и в количестве параметров функции.

Единственная функция, для которой не требуется прототип — это main(), так как это первая функция, вызываемая в начале работы программы.

**6,4Структура описание функции**

Сначала типо возвр знач функции-потом имя функции-затем открывем круглые скобки и перечисляем параметры,скобки закрываются открываются фигурные скобки и в них пишется реализация,после фиг скобок **;** не ставится

**6.5** есть адреса прибавляя 1 мы шагаем на один элемент, a int прибавляем 1 мы получается шагаем на размер inta в памяти впериод , то есть на 4 байта вперед

**6.6 int\*arr[N]- массив указателей**