**Защита лабораторной №2**

1. Данные в памяти абстрактной машины расположены друг за другом единым неразрывным блоком. К каждой ячейке можно обратиться по адресу. Адрес отвечает за доступ к ячейке. Может быть нецелым - NULL
2. Нулевой адрес – это адрес, указатель на который ничего не указывает.

Int \*a = NULL (такой указатель можно использовать при сравнении указателей). Не правдив.

1. Для определения адреса данных используется операция ‘&’ **взятия адреса.**
2. Зависит от архитектуры ОС. Размер адреса совпадает с размером машинного слова. Верхняя параша указывает на ячейку в памяти, которая имеет размер 1 байт, при этом сам размер адреса зависит от архитектуры.
3. На машине с адресом n байт можно адресовать 2^(n \* 8)

1 байт – 256 байт

2 байта – 64Кб

3 байта – 16 Мбайт

4 байта – 4 Гб

8 байта – 16 Экзобайта

1. **Тип size\_t** (беззнаковый) используется для адресных переменных, потому что он никогда не может превышать размер адреса, достаточного для адресации доступной оперативной памяти на машине с учетом архитектуры ОС
2. Минимальная единица адресации памяти компьютера – байт
3. **Указатель** – это объект, содержащий адрес объекта или функции

Указатели могут использоваться для передачи параметров в функцию, //обработки областей памяти

Разновидности указателей:

- типизированный указатель на данные (тип\* имя)

- бестиповой указатель (void\* имя)

- указатель на функцию

1. **Операция разыменования** (\* - астерикс) используется для доступа к данным, на которые указывает указатель. (указывает на ячейку в памяти)
2. Так int \*x, а не так int\* x? Потому что так нагляднее
3. новый адрес в p = старый адрес из p + n \* sizeof(тип)

новый адрес в p = старый адрес из p - m \* sizeof(тип)

pb – pa = (адрес из pb – адрес из pa) / sizeof(тип)

1. **Restrict** – это служебное слово в языке (начиная с версии c99), с помощью которого программист может указать компилятору, что указатель ссылается на область памяти, на которую больше никакой указатель не ссылается.

Это позволяет компилятору генерировать более оптимизированный код. Использование restrict определено только для указателей

1. **Висячий указатель**— [указатель](https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), не указывающий на допустимый объект соответствующего типа. Висячие указатели возникают тогда, когда объект удалён или перемещён без изменения значения указателя на нулевое, так что указатель все ещё указывает на область памяти, где ранее хранились данные.

**Утечка памяти** – неконтролируемое уменьшение свободной оперативной памяти. В цикле десять раз создаётся новый массив, а его адрес записывается в указатель. Адреса старых массивов при этом удаляются. Поэтому дальше оператор delete[] удаляет только последний созданный массив. Остальные останутся в памяти до тех пор, пока не будет закрыта программа

1. **Агрегированный тип данных** – тип данных, из сущности которого можно выделить другой определенный тип данных: простой или агрегированный.
2. Сущность агрегированного типа данных может включать сущность и другого, и того же агрегированного типа данных. (элементы массива – целые числа, элементы массива – массивы)
3. **Массив** – агрегированный тип данных, обладающий тремя свойствами:

- линейность – все элементы массива расположены в памяти по порядку единым неразрывным блоком

- однородность – все элементы одного типа и одного объема

- произвольный доступ – скорость доступа до любого элемента массива не зависит от его положения в массиве

1. Скорости доступа к элементам двух различных массивов могут отличаться.
2. **Статические одномерные массивы** — (массив с заранее известным количеством элементов) это комплект данных одного типа, располагающихся в *непрерывном* участке памяти так, что по индексу компонента определяется адрес его значения.
3. С помощью ключевого слова typedef можно определить новое имя уже существующего типа данных (typedef type1 type 2). Почему не макросы? Может, потому что заменяет ключевое слово
4. **Подпрограмма** – именованный участок кода.

**Точка входа** – строка, с которой начинается исполнение подпрограммы (В СИ она единственна)

**Точка выхода** – строка, которой заканчивается выполнение подпрограммы (В СИ может быть несколько) (строка, после которой управление передается в то место, откуда была вызвана функция)

1. **Явный возврат значения** – применение к значению оператора явного возврата. (оператор явного возврата в си – return)
2. **Функция** – подпрограмма, возвращающая значение явно.

**Процедура** - подпрограмма, не возвращающая значение явно.

1. **Формальные параметры** указываются при описании подпрограммы.

**Фактические параметры**– непосредственно при её вызове.

Фактические параметры часто называют *аргументами*.

1. В си процедуры рассматриваются как функции, которые ничего не возвращают. В них тип возвращаемого значения – void.
2. **Передача значения в функцию по значению** – создается локальная копия переданного аргумента.

**Передача по ссылке** – изменяется сама переданная сущность. В си нет передачи по ссылке.

1. **Область видимости (некоторой сущности)** – те строки кода, из которых можно получить доступ к этой сущности.

**Время жизни переменной** может быть либо "глобальным", либо "локальным". Объект с глобальным временем жизни характеризуется тем, что в течение всего времени выполнения программы с ним ассоциирована ячейка оперативной памяти и значение. Объекту с локальным временем жизни выделяется новая ячейка памяти при каждом входе в подпрограмму, в которой он определен или объявлен. Когда выполнение подпрограммы завершается, память, выделенная под локальный объект, освобождается и, следовательно, локальный объект теряет значение.

1. **Сигнатура** – запись определенного формата из типа возвращаемого значения, имени подпрограммы и списка аргументов.
2. В языках высокого уровня **описание** подпрограммы обычно состоит из двух частей: заголовка и тела.

Заголовок подпрограммы описывает её имя и параметры, т.е. информацию, необходимую для вызова подпрограммы (сигнатура)

Телоподпрограммы – набор операторов, который будет выполнен всякий раз, когда подпрограмма будет вызвана.

**Описание** – привязка имени к сущности

**Точка объявления** – строка кода, содержащая сигнатуру подпрограммы (инструкция компилятору, как использовать указанное имя)

**Точка вызова** – строка кода, в которой указывается имя полпрограммы и фактические параметры

1. Язык с возможностью перегрузки – язык, в котором в одной области видимости могут существовать подпрограммы с разными сигнатурами, но с одним и тем же именем. (в си с99 – без возможности перегрузки)
2. **Чистая функция** - это функция, которая является детерминированной и не обладает побочными эффектами.

Функция является **детерминированной,** если для одного и того же набора входных значений она возвращает одинаковый результат.

**Функции с побочными эффектами** – это функции, которые в процессе выполнения своих вычислений могут модифицировать значения глобальных переменных, осуществлять операции ввода/вывода.

1. **Рекурсивная функция** – функция, у которой в теле находится точка ее же вызова.

**База рекурсии** – набор условий, определяющий окончание рекурсивных обращений

Если функция f(n) может вернуть некоторое значение для любого n из множества N, встав на базу, то база является **полной** для множества N.

1. **Глубина рекурсии** ограничена, так как может быть переполнен стек вызовов функции. (Стек вызовов – может являться как частью процессора, так и ОС, организован по типу стека и в нем хранятся все вызовы функции)

**Хвостовая рекурсия** – вид рекурсии, в котором рекурсивный вызов является последней инструкцией перед точкой выхода.

*Дополнительно:*

Когда параметр передается по значению, значение фактического параметра используется для инициализации соответствующего формального параметра, который в дальнейшем действует как локальная переменная. Метод передачи по ссылке передает путь доступа к данным (обычно – просто адрес) в вызываемую подпрограмму. Это открывает доступ к ячейке памяти, хранящий фактический параметр. Таким образом, вызываемая подпрограмма может получить доступ к фактическому параметру.

**Объявление** – это инструкция компилятору, как использовать указанное имя (описывает свойства переменной или функции). Объявлений одного и того же имени может быть сколько угодно, главное чтобы они все были согласованы (т.е. одинаковы).

**Определение** - осуществляет привязку имени к сущности (к памяти для данных или к коду для функций), т.е. специфицирует код или данные, которые стоят за этим именем. В языке Си существует правило единственного определения. Это означает, что определение может быть только одно.