Лекция №18
по дисциплине
«ТЕОРИЯ АЛГОРИТМОВ»

УКАЗАТЕЛЬ

Преподаватель:
Золотоверх Д.О.

ХРАНЕНИЕ ДАННЫХ В (ОЗУ)

В каждой ячейке хранится 8 бит (8 бит = 1 байт).

У каждой ячейки есть адрес.

Адрес всегда указывается как целое число (зачастую в шестнадцатеричном формате).

Адрес, в зависимости от архитектуры может быть 8, 16, 32 или 64 битным.

АДРЕС

Каждая переменная имеет адрес.

Этим адресом является ячейка памяти, в которой содержится значение переменной.

У нас нет возможности выбирать адрес и при многоразовом запуске программы он может меняться.

Адресацией занимается операционная система.

СИНТАКСИС

Чтобы узнать адрес переменной необходимо использовать символ &.

Такой символ называется амперсанд.

Символ ставится перед переменной, адрес которой необходимо узнать:

```
&названиеПеременной;
```

Тип адреса является целым числом, но не совсем.

```
#include <iostream>

using namespace std;

int main () {
    int number = 228;
    printf(
       "Mem. addr.: %X",
       &number
    );
}
```

ХРАНЕНИЕ АДРЕСА

Если попытаться запомнить адрес с помощью переменной, не один тип данных, что был использован нами ранее, не подойдет.

Тип данных, что нужен для хранения данных в таком виде называется указатель.

В отличии от других языков программирования, указатель именно характерен для C и C++. Его использование и дает высокую скорость работы программы.

УКАЗАТЕЛЬ

Указатель (pointer) – это переменная, диапазон значений которой состоит из существующих адресов ячеек памяти или специального значения – нулевого адреса.

Указатель, имеет тип, НО этот тип относится к переменной, на которую он «указывает».

Сам по себе указатель ВСЕГДА является целым числом (все адреса памяти – целые числа, зачастую изображены в шестнадцатеричном формате.

СИНТАКСИС

Как говорилось ранее, указатель имеет тип, он должен быть такого же типа, как и переменная, на которую он указывает*.

Синтаксис:

```
тип_данных *названиеУказателя;
тип_данных * названиеУказателя;
тип_данных* названиеУказателя;
```

Желательно использовать первый способ.

```
#include <iostream>


using namespace std;

int main() {
    int number = 228;
    int *ptr = &number;
    printf(
        "Mem. addr.: %p",
        ptr
    );
}
```

ХРАНЕНИЕ АДРЕСА УКАЗАТЕЛЕМ

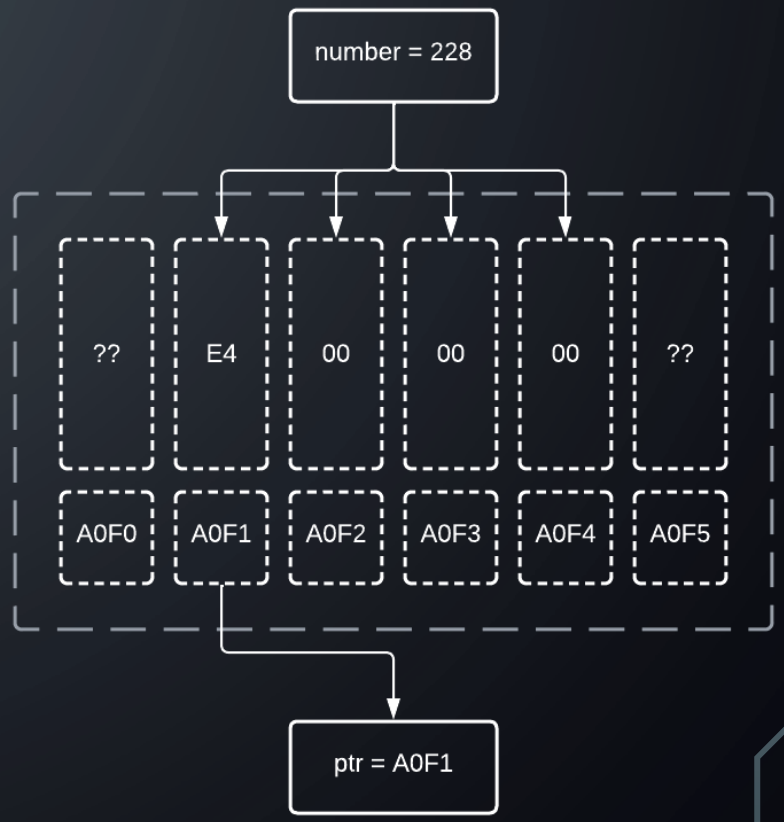
Тип указателя нужен для того, чтобы знать как хранятся данные и сколько им нужно места.

Сам по себе указатель не нуждается в корректном типе, но данные, при их удалении, модификации и считывании – наоборот.

Также следует учесть порядок расположения байтов. Существуют:

- от старшего к младшему (big-endian);
- от младшего к старшему (little-endian).

Пример использует архитектуру x86 (little-endian).



РАЗЫМЕНОВАНИЕ УКАЗАТЕЛЯ

Чтобы из указателя обратно получить данные, необходимо использовать оператор разыменования.

Синтаксис:

```
*названиеУказателя;
```

Не стоит путать с созданием:

```
тип_данных *названиеУказателя;
```

Их отличие в том, что создание использует тип.

```
int main() {
    int number = 228;
    int *pointer = &number;
    printf(
        "Mem. addr.: %p\n",
        pointer
    );

    *pointer = 1337;
    printf(
        "New value: %d\n",
        number
    );
}
```

УКАЗАТЕЛЬ УКАЗАТЕЛЯ

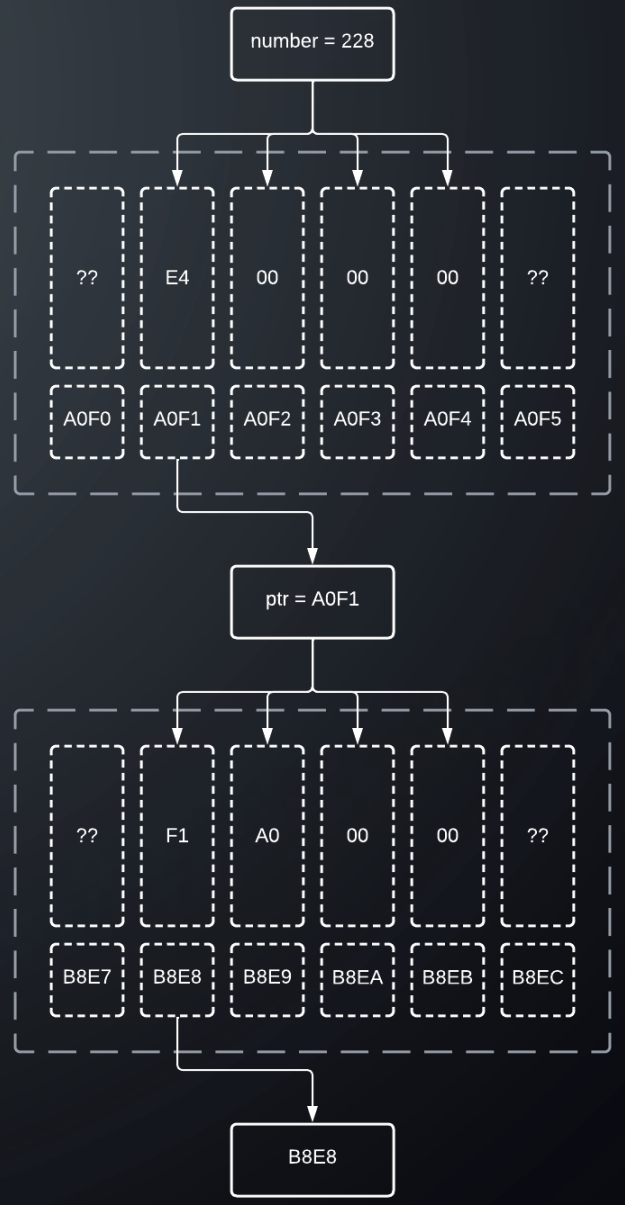
Указатель может указывать на другой указатель. В таком случае он хранит адрес того указателя, который хранит адрес значения.

Синтаксис:

тип_указателя **название указателя;

Пример:

```
int number = 228;
int *ptr = &number;
int **pptr = &ptr;
```



СПАСИБО ЗА ВНИМАНИЕ!