```cpp
#include <iostream>
using namespace std;

int main()
{
    cout<<"1d char array"<<endl;
    char name[] = "Ibtasaam Amjad"; //15 characters including '\0'

    cout<<name<<endl; //Ibtasaam Amjad, it's a pointer to the char
array so it outputs Ibtasaam Amjad
    cout<<&name<<endl; // pointer to pointer to 1d array so it
outputs Address

    cout<<*name<<endl; //dereferencing it decays the pointer to the
array to a pointer to a character. so it outputs 'I'
    cout<<name + 1<<endl; // now it points to 'b' but it is not
dereferenced so it outputs until '\0' is encountered.
    cout<<*(name + 1)<<endl; // now it's outputting 'b'
    cout<<*name + 1<<endl; // adds 1 to ascii value of 'I' so it
outputs 74

    cout<<(unsigned long long)(&name)<<endl;
    cout<<(unsigned long long)(&name + 1)<<endl; // diff is 15 bytes
because the array is of size 15 and we are adding 1 to the pointer
to the pointer resulting in deviation of 15 bytes

    cout<<endl<<"2d character array"<<endl;
    char names[5][10] = {"Ibtasaam", "Omar", "Aryan", "Maaz",
"Rayyan"};

    cout<<names<<endl; //pointer to the 2d array
    cout<<&names<<endl; //pounter to pointer....outputs Address
    cout<<*names<<endl; //derefencing it outputs the first row
    cout<<*(names + 1)<<endl; //moves 1 row over and outputs Omar
    cout<<((names + 2) + 3)<<endl; //moves 2 rows over and 3 cols
over in that specific row

    cout<<(unsigned long long)(&names)<<endl;
    cout<<(unsigned long long)(&names + 1)<<endl; //now we are
adding 1 to the pointer to pointer to the 2d array so it will
deviate by 50 bytes because it's size is 50 bytes
}

memory understanding for character arrays
```

1D Character Array

cpp

Copy code

```
char name[] = "Ibtasaam Amjad"; // 15 characters including the null terminator ('\0')
```

- A 1D character array (name) is initialized with the string "Ibtasaam Amjad". Strings in C++ are null-terminated, so the array has 15 characters in total (14 visible characters + 1 null terminator).

Pointer Behavior in a 1D Character Array

cpp

Copy code

```
cout<<name<<endl;
```

- name decays to a pointer to the first character. When you use cout with a char*, it treats it as a C-style string and prints the entire string until the null terminator (\0).

cpp

Copy code

```
cout<<&name<<endl;
```

- &name is a pointer to the entire array. It points to the array itself and outputs the address of the array.

cpp

Copy code

```
cout<<*name<<endl;
```

- Dereferencing name (*name) gives the first character of the array, which is 'I'.

cpp

Copy code

```
cout<<name + 1<<endl;
```

- name + 1 advances the pointer to the second character ('b'). When printed, it outputs the string starting from 'b' ("btasaam Amjad").

cpp

Copy code

```
cout<<*(name + 1)<<endl;
```

- Dereferencing name + 1 (*(name + 1)) gives the second character, 'b'.

cpp

Copy code

cout<<*name + 1<<endl;

- *name is 'I'. Adding 1 to its ASCII value (73) results in 74, which corresponds to the character 'J'. The output is 74.

Pointer Arithmetic in a 1D Character Array

cpp

Copy code

cout<<(unsigned long long)(&name)<<endl;

cout<<(unsigned long long)(&name + 1)<<endl;

- &name points to the array as a whole.
- &name + 1 moves the pointer by the size of the entire array (15 bytes, as the array contains 15 characters).
- The difference in memory addresses between &name and &name + 1 is **15 bytes**.

2D Character Array

cpp

Copy code

char names[5][10] = {"Ibtasaam", "Omar", "Aryan", "Maaz", "Rayyan"};

- A 2D array names is declared with 5 rows and 10 columns. Each row can store up to 9 visible characters plus the null terminator (\0). The rows are initialized with different strings.

Pointer Behavior in a 2D Character Array

cpp

Copy code

cout<<names<<endl;

- names is a pointer to the first row of the array (names[0]). However, because the row itself is a character array, it prints the first row ("Ibtasaam") when used with cout.

cpp

Copy code

cout<<&names<<endl;

- &names is a pointer to the entire 2D array. It outputs the address of the entire array.

cpp

Copy code

cout<<*names<<endl;

- Dereferencing names (*names) gives the first row (names[0]), which is a character array. When printed, it outputs "Ibtasaam".

cpp

Copy code

cout<<*(names + 1)<<endl;

- names + 1 moves the pointer to the second row (names[1], which contains "Omar"). Dereferencing it (*(names + 1)) gives "Omar".

cpp

Copy code

cout<<((names + 2) + 3)<<endl;

- names + 2 moves the pointer to the third row (names[2], which contains "Aryan").
- Adding 3 moves the pointer 3 columns over in the third row, pointing to the fourth character ('a' in "Aryan").

Pointer Arithmetic in a 2D Character Array

cpp

Copy code

cout<<(unsigned long long)(&names)<<endl;

cout<<(unsigned long long)(&names + 1)<<endl;

- &names is a pointer to the entire 2D array.
- &names + 1 moves the pointer by the size of the entire 2D array (5 rows * 10 columns = 50 bytes).
- The difference in memory addresses is **50 bytes**.

Key Concepts from the Program

1. **Null-Terminated Strings:**
    - A string in C++ is stored as a character array with a null terminator ('\0'), which marks the end of the string.
2. **Pointers and Arrays:**

- In C++, the name of an array decays into a pointer to its first element. For example:
  - name (1D array) is a pointer to the first character.
  - names (2D array) is a pointer to the first row.

3. **Pointer Arithmetic:**
   - Adding 1 to a pointer advances it by the size of the type it points to:
     - For char, this is 1 byte.
     - For a 1D array, advancing a pointer moves it by the size of the array's type.
     - For a 2D array, adding 1 to a row pointer moves it to the next row.
4. **Memory Layout of 2D Arrays:**
   - A 2D array is stored in a **row-major order**, meaning all elements of the first row are stored contiguously in memory, followed by the elements of the second row, and so on.
5. **Difference Between name and &name:**
   - name decays to a pointer to the first element (or row).
   - &name is a pointer to the entire array.
6. **Dereferencing in Character Arrays:**
   - Dereferencing (*) in a 1D array gives the first character.
   - Dereferencing in a 2D array gives the first row (a character array).