PRACTICAL NO. 2

AIM: a)write a program to define structure and pointers.

b) what is mallac/calloc . Explain with a suitable example.

---

a)write a program to define structure and pointers.

THEORY:

In this program, we define a structure Person with two members: a string name and an integer age. In the main() function, we create a variable person of type Person and initialize its name and age members.

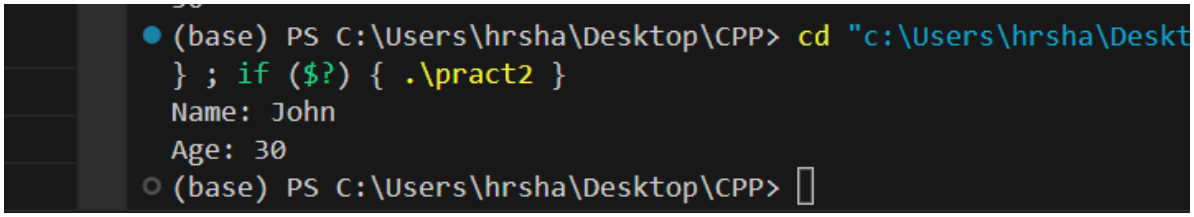
Next, we declare a pointer personPtr of type Person*, which we will use to point to the person variable. We assign the address of person to personPtr using the & operator.

Finally, we use the arrow operator -> to access the name and age members of the person variable through the pointer personPtr. The -> operator is used to dereference the pointer and access its members.

Code:

```
#include <iostream>

using namespace std;

// Define a structure for a person

struct Person {

   string name;

   int age;

};

int main() {
```

```
    // Declare a person and initialize their name and age
    Person person;
    person.name = "John";
    person.age = 30;
    Person *personPtr;
     personPtr = &person;
     cout << "Name: " << personPtr->name << endl;
    cout << "Age: " << personPtr->age << endl;
    return 0;
}
```



---

b) what is mallac/calloc . Explain with a suitable example.

Theory

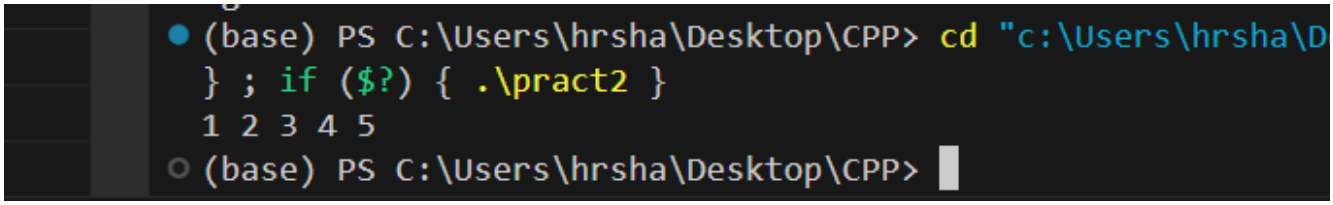
malloc() and calloc() are two memory allocation functions in C++. They are used to dynamically allocate memory at runtime. malloc() function: The malloc() function is used to allocate a block of memory of specified size in bytes.

It takes a single argument that specifies the number of bytes to be allocated. The return value of the malloc() function is a pointer to the first byte of the allocated block. If the allocation fails, malloc() returns a null pointer.

Code:

```
#include <iostream>
#include <cstdlib>
```

```
using namespace std;

int main() {

    int *ptr;

    int n = 5;

    ptr = (int*) malloc(n * sizeof(int));

    if (ptr == NULL) {

        cout << "Memory allocation failed" << endl;

        exit(1);

    }

    for (int i = 0; i < n; i++) {

        *(ptr + i) = i + 1;

    }

    for (int i = 0; i < n; i++) {

        cout << *(ptr + i) << " ";

   }

    free(ptr);

    return 0;

}
```



CONCLUSION:

Thus we have successfully executed programs .