Techimax

Fundamentals of Programming in C++

File I/O

```
#include <fstream>
```

```
ios::in
Input Mode
Open a file for reading.
```

```
ios::out
Output Mode
Open a file for writing.
```

```
ios::app
Append mode.
All output to that file to be appended to the end.
```

```
ios::trunc
Truncate Mode
If the file already exists,
its contents will be truncated before opening the file.
```

File I/O

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

int main()
{
    ofstream myFile1("file1.txt");
    myFile1 << "This file has been written using a fstream.\n";
    myFile1.close();
}
```

File I/O

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

int main()
{
    //**Appending lines
    ofstream myFile2("file2.txt", ios::app);
    if (!myFile2)
    {
        cout << "Sorry the file could not be opened!!.\n";
    }

    myFile2 << "Lines get appended." << endl;
    myFile2.close();
}
```

File I/O

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

int main()
{
    // ** Reading from a file
    ifstream myFile3;
    myFile3.open("file3.txt");
    string data;
    while (myFile3)
    {
        getline(myFile3, data);
        cout << data << endl;
    }
}
```

```
#include <string>
#include <cstring>
```

Strings

```
char chArray[] = "chArray";
string str = "string";
```

Character Array

- A character array is simply an **array of characters** can terminated by a null character.
- Size of the character array has to **allocated statically**, more memory cannot be allocated at run time if required. Unused allocated **memory is wasted** in case of character array.
- Implementation of **character array is faster** than std:: string.

Strings

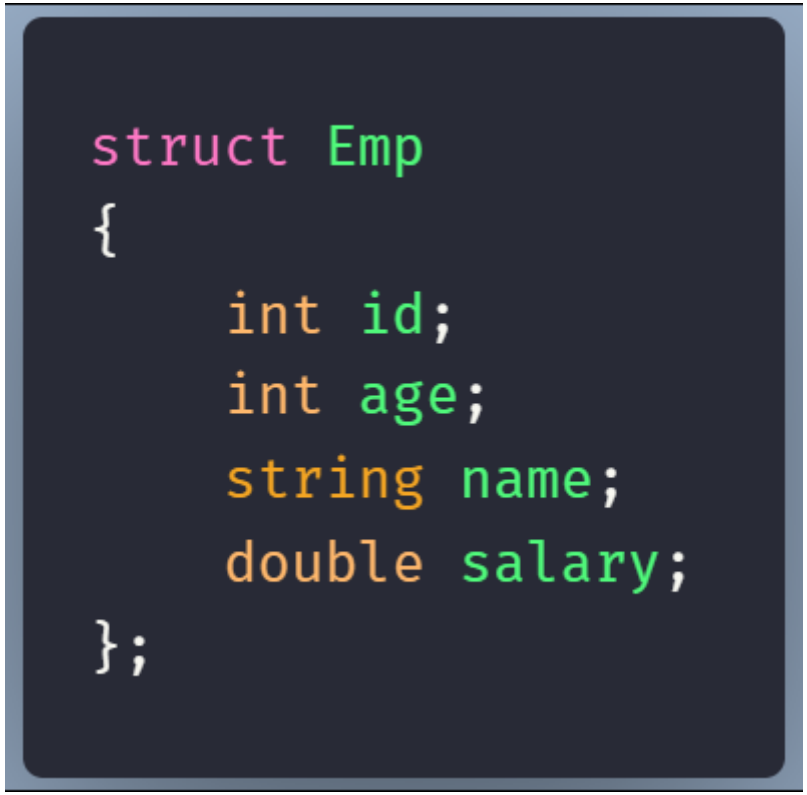
- A string is a **class which defines objects** that be represented as stream of characters.
- In case of strings, memory is **allocated dynamically**. More memory can be allocated at run time on demand. As no memory is pre-allocated, **no memory is wasted**.
- **Strings are slower** when compared to implementation than character array.

Strings

```
s1.push_back('1');
s2.pop_back();
s1.swap(s2);
string s3 = s1.append(s2);
string s4 = s1 + s2;
cout << s1.length() << " " << s3.size() << endl
```

Structures

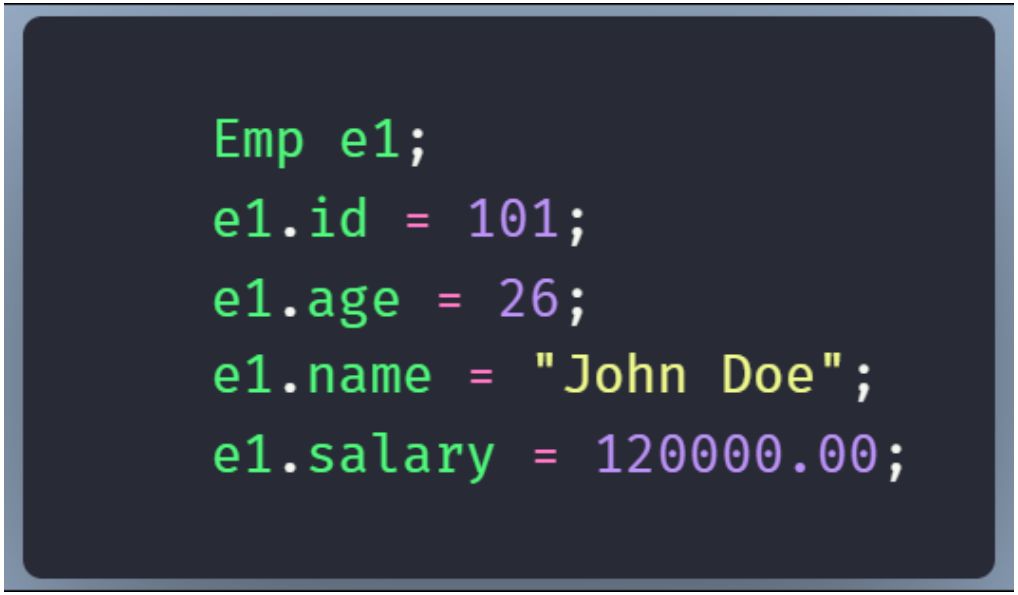
Structures in C++ are user defined data types which are used to store group of items of non-similar data types.



```
struct Emp
{
    int id;
    int age;
    string name;
    double salary;
};
```

Structures

Declare and assign Values



Structures

Array of Structures

```
Emp empAr[20];
empAr[0].id = 101;
empAr[0].name = "Tanmay Bhat";
empAr[0].age = 28;
empAr[0].salary = 120000.0;

empAr[1] = {102, 30, "Zakir Khan", 200000.0};
```

Structures

Structures and Pointers

```
struct Point
{
    int x, y;
};

int main()
{
    struct Point p1 = {1, 2};
    // p2 is a pointer to structure p1
    struct Point *p2 = &p1;
    // Accessing structure members using
    // structure pointer
    cout << p1.x << " " << p1.y << endl;
    cout << p2->x << " " << p2->y;
    return 0;
}
```