27-06-2024

Differences between malloc() & new

malloc()	new
SYNTAX:-	SYNTAX:-
(int *) malloc(sizeof(int))	new int
-> malloc cannot carry calculate size of the datatype, so it needs sizeof operator to calculate size of datatype	-> new automatically calculates size of datatype, so no need of size of operator
-> malloc returns void pointer so it needs Typecasting	-> new returns current type pointer so no need of Typecasting
-> It's a function() in C	-> It is an operator

Ex:-

```
Using namespace std;
#include <iostream>
main()
{
    int *p;
    p = new int;
    cout << "enter n: ";
    cin >> *p;
    cout << *p;
}
```

Q:- Add of Complex numbers

```
-> using namespace std;
#include<iostream>
class complex
{
   private: int a,b;
   public: void get()
           {
              cout << "enter a,b:";
              cin >> a>b;
           }
           void add(complex *p, complex *q)
           {
            a=p->a+q->a;
            b=p->b+q->b;
           }
           void display()
           {
             cout << a << "+i" <<b;
           }
};
main()
{
   complex *p,*q,*r;
   p= new complex();
   q= new complex();
   r= new complex();
   p->get(); q->get(); r->add(p,q); r->display();
}
```

p -> object
q -> argument
r -> local variable

abcd.cpp

OPERATOR OVERLOADING =>

```
-> using namespace std;
   #include <iostream>
   class complex
   {
     private: int a, b;
     public : void get ()
              {
                cout << "enter a, b : ";
                cin >> a >> b;
              }
              complex operator + (complex q)
             {
               complex r;
               r.a = a+q.a ; r.b = b + q.b;
               return r;
             }
              void display ()
              {
                cout << a << "+i" << b;
              }
   };
   main ()
   {
     complex p, q, r;
     p.get () ; q.get () ; r = p + q;
     r. display ();
     int a = 5 + 2;
     cout << a;
   }
```