Q1. Explain Polymorphism in C++ Programming?
Ans: Polymorphism in C++ Programming is fundamental concept of object oriented Programming that allows functions or methods behave different based on the context, typically the type of object or data involved

Example: + is an overloaded operator in C++
If Operands are string then it will work as a string concationato operators

```
string s1="hello"
string s2="world"
string s3=s1+s2;
```

Example:
If both operands are numeric

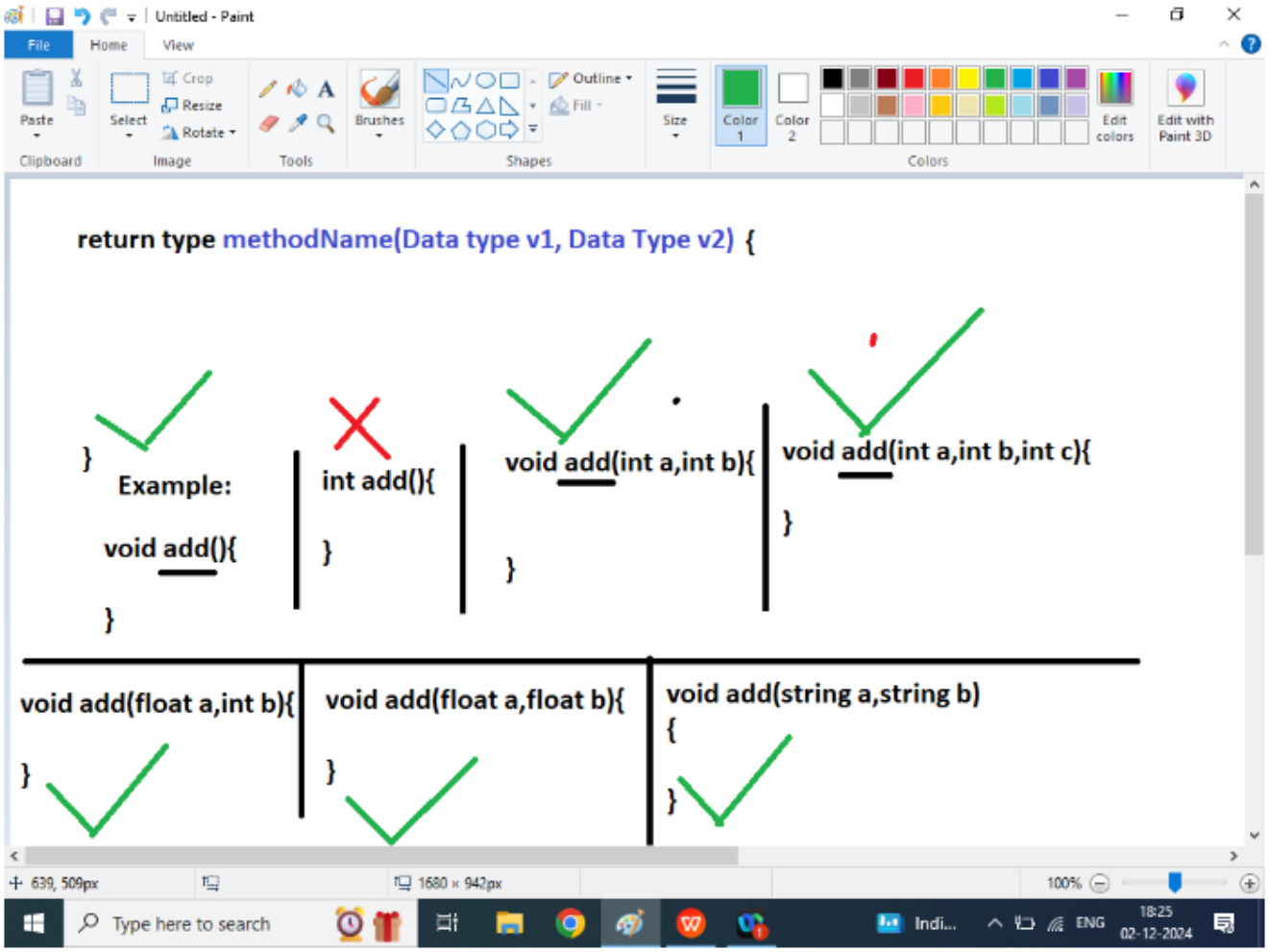
```
int A,B,C;
A=10;
B=20;
C=A+B;
```

Poly means many forms
One Name Different form is known as Polymorphism

Types of Polymorphism
1. Compile time Polymorphism/ Static Binding
Example: method Overloading, Operator overloading
2. Run time Polymorphism / Dynamic Binding
Example: Method Overriding (virtual function)
Compile time Polymorphism/ Static Binding :

The compile time Polymorphism achieved by method overloading and operator overloading. The behavior of the function or operator is determined a compile time

Q2. Explain Method Overloading in c++?
Ans:
This allows multiple functions to have same name but different parameters list [No of Parameters , Data types of Parameters]



// Method Overloading

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

class Test{
public:
void add(){
int a,b;
a=1;
b=2;
cout<<"\nAddition without argument "<<(a+b);
}
void add(int a,int b){//formal Argument
cout<<"\nAddition with two int argument "<<(a+b);
}

void add(float a,int b){//formal Argument
cout<<"\nAddition with two (float,int) argument "<<(a+b);
}
void add(float a,float b){//formal Argument
cout<<"\nAddition with two float  argument "<<(a+b);
}
void add(string a,string b){//formal Argument
cout<<"\nAddition with two  argument "<<(a+b);
}
};
int main(){
Test t;
t.add();
t.add(2.5f,6);
t.add(1.5f,9.5f);
t.add("10","20");
return 0;
```

```
}


// Method Overloading
#include<iostream>
using namespace std;

class Test{
public:
void add(){
int a,b;
a=1;
b=2;
cout<<"\nAddition without argument "<<(a+b);
}
void add(int a,int b){//formal Argument
cout<<"\nAddition with two int argument "<<(a+b);
}

void add(float a,int b){//formal Argument
cout<<"\nAddition with two (float,int) argument "<<(a+b);
}
void add(float a,float b){//formal Argument
cout<<"\nAddition with two float  argument "<<(a+b);
}
void add(string a,string b){//formal Argument
cout<<"\nAddition with two  argument "<<(stoi(a)+stoi(b));
}
};
int main(){
Test t;
t.add();
```

```
t.add(2.5f,6);
t.add(1.5f,9.5f);
t.add("10","20");
return 0;
}
```

Q2. Explain Operator Overloading in C++ ?

Ans: Operator overloading in C++ is a features that allows to redefine the behaviour of operators

1. Operators can be overload with the help of member functions
2. The syntax and precedence of the operator cannot be change
3. Use operator keyword to overload a operators
4. We can overload only Pre-defined operators in c++. we cannot create any new operators
5. We cannot overload following operators in C++[:: , sizeof, .,->]

```
// operaor Overloading
#include<iostream>
using namespace std;

class Space{
public:
int x;
int y;
int z;

Space(int x,int y,int z){
this->x=x;
this->y=y;
```

```
this->z=z;
}
void showData(){
cout<<"\n (x,y,z) :("<<x<<", "<<y<<","<<z<<")";
}
void operator -(){
x=-x;
y=-y;
z=-z;
}

};
int main(){
Space s(10,-20,30);
s.showData();
-s;
s.showData();
return 0;
}
```

Next class Topic: Pointers