Lambda Expressions

lambda functions -> small, we can write this function directly in the code.
usefull when quick function without naming it /declaring it separately

"mini function"

```
[capture] (parameters) -> return_type
{
    code
};
```

Basic Example:

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

int main()
{
  auto msg = []()
     {
        cout<<"hello  I am from Bangalore!!"<<endl;
     };

   msg();
  return 0;
}
```

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

int main()
{
  auto msg = []()
     {
        cout<<"hello  I am from Bangalore!!"<<endl;
     };

   auto add = [] (int a,int b) -> int
   {
     return a+b;
```

```
   };

   msg();

   cout<<"Additon is : "<<add(5,10);

  return 0;
}
==================================
#include<iostream>
using namespace std;

int main()
{
  auto message = []()
     {
        cout<<"Hello Rajesh!!"<<endl;
     };

   message();
  return 0;
}

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

int main()
{
  auto sub = [](int a, int b) -> int
     {
        return a-b;
     };

   cout<<sub(10,3);
  return 0;
}
=======================================

#include<iostream>
#include<functional> // function with lambda
using namespace std;
//Passing lambda to functions
```

```
void print(function <void ()> kavya )  //function taking input as lambda
{
   for(int i=0;i<10;i++)
   {
      kavya();
   }
   //kavya();
   //kavya();
}

int main()
{
  auto message = []()
    {
       cout<<"Hello Students!!"<<endl;
    };

   print(message);// function taking input as lambda function
  return 0;
}
================================================

#include<iostream>
using namespace std;
//using lambdas in loops

int main()
{
   for(int i=0;i<5;i++)
   {
      auto message = [i]()
       {
        cout<<"Hello Students!! --> "<<i<<endl;
       };

       message();
   }
  return 0;
}
==================================
#include<iostream>
using namespace std;
```

```
//passing value in capture clause

int main()
{
   int x = 250,y=180;

      auto show = [y]() //capture clause []
       {
        cout<<y<<endl;
       };

       show();

  return 0;
}
============================================================

#include<iostream>
using namespace std;
//passing value in capture clause

int main()
{
   int x = 250,y=180;

      auto show = [y]() //capture clause []
       {
        cout<<y<<endl;
       };

       y=410;
       show(); //180

  return 0;
}

================================================

#include<iostream>
using namespace std;
//passing reference in capture clause

int main()
```

```
{
   int x = 250,y=180;

      auto show = [&y]() //capture clause []
       {
        cout<<y<<endl;
       };

       y=410;
       show();//410

  return 0;
}
```

==========

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

int num=1000;

int main()
{
   int x = 250,y = 120, z=850;

      auto show = [=]() //capture by value
       {
        //cout<<z<<endl;//fine
        cout<<x<<" "<<y<<" "<<z<<" "<<num<<endl;
       };

       show();

  return 0;
}
```

==========

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

int num=1000;

int main()
```

```
{
    int x = 250,y = 120, z=850;
    char ch='K';
    double db=32.5;
    string name="Bangalore";

       auto show = [=]() //capture by value
        {
         //cout<<z<<endl;//fine
         cout<<x<<" "<<y<<" "<<z<<" "<<num<<endl;
         cout<<ch<<" " <<db<< " "<<name<<endl;
        };

        show();

  return 0;
}

====================================
#include<iostream>
using namespace std;
//specific capture

int main()
{
    int x = 250,y = 120;

       auto show = [x,&y]() //capture by value
        {
          cout<<x<<" "<<y<<endl;
        };

        x=800;
        y=950;

        show();

  return 0;
}

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

```
//specific capture

int main()
{
   int x = 250,y = 120;

      auto show = [x,&y]() //capture by value
       {
          //x++; //not possible bcz x  has only value
          y++;
         cout<<x<<" "<<y<<endl;
       };
       show();

  return 0;
}

================================

#include<iostream>
using namespace std;
//specific capture

int main()
{
   int x = 250,y = 120;

      auto show = [x,&y]() mutable
       {
          x++; // possible bcz lambda is mutable
          y++;
         cout<<x<<" "<<y<<endl;
       };
       show();

  return 0;
}

================================================
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
```

```
int main()
{
   vector<int> nums = {5,2,9,1};

   sort(nums.begin(),nums.end(),[](int a,int b)
   {
       return a>b; //desc  , a<b ascen
   });

   for(int x : nums)
   {
       cout<<x<<" ";
   }
   cout<<endl;
  return 0;
}

==================================
#include<iostream>
using namespace std;
//normal function pointer

void print()
{
    cout<<"hello"<<endl;
}

int main()
{
  void (*ptr)() = print;
  ptr();
  return 0;
}

==================================
#include<iostream>
using namespace std;
//lamdba function pointer ==> not possible

int main()
{
    int a =10;
```

```
    auto str = [a]()
    {
        cout<<a<<endl;
    };

  void (*ptr)() = str;

  ptr();
  return 0;
}

==========================================
#include<iostream>
using namespace std;
//lamdba function pointer

int main()
{
    auto str = []() //without capturing
    {
        cout<<"take care"<<endl;
    };

  void (*ptr)() = str;

  ptr();
  return 0;
}

==========================================
#include<iostream>
using namespace std;
//generic lambdas

int main()
{
    auto var = [](auto x) //without capturing
    {
        cout<<x<<endl;
    };

   var(10);
   var("Pawan");
```

```
   var(22.3);
   var('l');

  return 0;
}
```