```
a ? 2
b ? 3
c ? 25
 . . . now doubling everything . . .
a=4 b=6 c=50
```

Program with "main" function only.

```
#include<iostream>
using namespace std;
int main()
{int a,b,c;
 cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 cout<<"\n . . . now doubling everything . . .";
 a=a*2;
 b=b*2;
 c=c*2;
 cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
 return 0;
}
```

Program with "main" function that calls function "DoubleThings". Three parameters are passed to DoubleThings and the results are printed in DoubleThings.

No "Function Prototype" is used. Instead, function is put before main.

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

void DoubleThings(int a , int b , int c)
      { a = a * 2 ;
        b = b * 2 ;
        c = c * 2 ;
        cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
      }

int main()
{int a,b,c;
 cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 DoubleThings(a,b,c) ; // call function
 return 0;
}
```

Program with "main" function that calls function "DoubleThings". Three parameters are passed to DoubleThings and the results are printed in DoubleThings.

Here, a Function Prototype *is* used.

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

void DoubleThings(int , int , int);

int main()
{int a,b,c;
 cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 DoubleThings(a,b,c) ; // call function
 cout<<"\n\n";
 return 0;
}

void DoubleThings(int a , int b , int c)
     { a = a * 2 ;
       b = b * 2 ;
       c = c * 2 ;
       cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
     }
```

Program with "main" function that calls function "AddNumbers".

If we have just <u>one</u> thing to return to main we can give the function itself a type (in this case int) and then use a return statement to send a value back.

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

int AddNumbers(int , int , int );

int main()
{int a , b , c ;
 cout<<"\n a ? ";
 cin>> a ;
 cout<<"\n b? ";
 cin>> b ;
 cout<<"\n c? ";
 cin>> c ;
 cout<<"\n\n sum="<< AddNumbers(a,b,c) ;
 return 0;
}

int AddNumbers(int  a, int b, int c)
      {int sum ;
       sum = a + b + c ;
       return sum;
      }
```

Note the "scope" of variables is usually limited to where they are declared. If a variable of the same name is declared elsewhere, it is a DIFFERENT variable, even though it has the "same name".

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

void DoubleThings(int , int , int);

int main()
{int a,b,c;
 cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 DoubleThings(a,b,c) ;
 cout<<"\n\n";
 return 0;
}

void DoubleThings(int a , int b , int c)
     { a = a * 2 ;
       b = b * 2 ;
       c = c * 2 ;
       cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
     }
```

a_{main}

$a_{DoubleThings}$

so, if you had put the line that does the printing:

```
cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
```

back in main, it would have printed the original (unchanged) values because $a_{DoubleThings}$ is changing - not a_{main}

Program with "main" function that calls function "DoubleThings". Three parameters are passed to DoubleThings and the results are printed after returning back to main. Global variables are used to accomplish this.

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

int aGlobal , bGlobal , cGlobal ;  // "global" variables

void DoubleThings(int , int , int);

int main()
{int a,b,c;
 cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 DoubleThings(a,b,c) ; // call function
 cout<<"\n\na="<<aGlobal<<" b="<<bGlobal<<" c="<<cGlobal;
 return 0;
}

void DoubleThings(int a , int b , int c)
     { aGlobal = a * 2 ;
       bGlobal = b * 2 ;
       cGlobal = c * 2 ;
     }
```

Program with "main" function that calls function "DoubleThings". Three parameters are passed to DoubleThings and the results are printed after returning back to main. Global variables and Global Scope Operator ("::") are used to accomplish this.

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

int a , b , c ;  // "global" variables

void DoubleThings(int , int , int);

int main()
{cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 DoubleThings(a,b,c) ; // call function
 cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
 cout<<"\n\n";
 return 0;
}

void DoubleThings(int a , int b , int c)
      { ::a = a * 2 ;
        ::b = b * 2 ;
        ::c = c * 2 ;
      }
```

Program with "main" function that calls function "DoubleThings". Three parameters are passed by REFERENCE (instead of by VALUE as in all previous examples). Now any changes to variables in the Function actually change variables in the main function that called it.

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

void DoubleThings(int & , int & , int & );

int main()
{int a,b,c;
 cout<<"\na ? ";
 cin>>a;
 cout<<"\nb ? ";
 cin>>b;
 cout<<"\nc ? ";
 cin>>c;
 DoubleThings(a,b,c) ; // call function
 cout<<"\n\na="<<a<<" b="<<b<<" c="<<c;
 return 0;
}

void DoubleThings(int & a , int & b , int & c)
      { a = a * 2 ;
        b = b * 2 ;
        c = c * 2 ;
      }
```