Functions

- Program definition

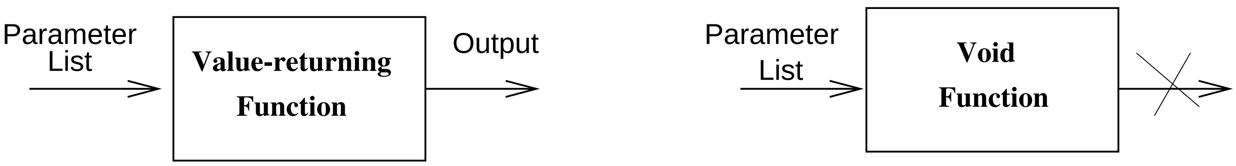
  - Programs are made up of functions (one of them has to be *main*)

```
Declaration
  .
  .
  .
FunctionDefinition
FunctionDefinition
  .
  .
  .
```

- Function Definition

```
DataType FunctionName ( ParameterList )
{
Statement
  .
  .
  .
}
```

- Value-returning vs void functions



Syntax Template

```
DataType FunctionName (ParameterList)
{
 Statement

    ...

}
```

Syntax Template

```
void  FunctionName (ParameterList)
{
 Statement

    ...

}
```

```
#include <iostream.h>
#include <math.h>

int compute_sum(int,int);

int main()
{
 int x,y,sum;

 cout << "Enter x and y";
 cin >> x >> y;

 sum=compute_sum(x,y);

 sum=2.0*sqrt(sum)-1.0;
 cout << sum;

 return 0;
}

int compute_sum(int num1, int num2)
{
  int result;

  result=num1 + num2;

  return result;
}
```

function prototype (declaration)

function definition

can be called from within an expression

```
#include <iostream.h>

void print_message();

int main()
{
 print_message();
 return 0;
}

void print_message()
{
 cout << "Hello World" << endl;
}
```

stand-alone statement

- Why are we using functions ?

  - To make programs easier to understand

  - To avoid repetitive coding

```
#include <iostream.h>

int main()
{
  cout << "**************************" << endl;
  cout << "**************************" << endl;
  cout << "   Welcome Home      "  << endl;
  cout << "**************************" << endl;
  cout << "**************************" << endl;
  cout << "**************************" << endl;
  cout << "**************************" << endl;
 return 0;
}
```

```
#include <iostream.h>

void Print2Lines();
void Print4Lines();

main()
{
  Print2Lines();
  cout << "   Welcome Home      "  << endl;
  Print4Lines();
  return 0;
}

void Print2Lines();
{
  cout << "**************************" << endl;
  cout << "**************************" << endl;
}

void Print4Lines();
{
  cout << "**************************" << endl;
  cout << "**************************" << endl;
  cout << "**************************" << endl;
  cout << "**************************" << endl;
}
```

More general !!!

```
#include <iostream.h>
void PrintLines(int);

int main()
{
 PrintLines(2);
 cout << "    Welcome Home    " << endl;
 PrintLines(4);
 return 0;
}

void PrintLines(int n)
{
  int i;

  for(i=0; i<=n i++)
     cout << "********************************" << endl;
}
```

Even more efficient !!!

```
#include <iostream.h>
void PrintLines(int,char);

int main()
{
 PrintLines(2,'*');
 cout << "    Welcome Home    " << endl;
 PrintLines(4,'#');
 return 0;
}

void PrintLines(int n, char ch)
{
  int i,j;

  for(i=0; i<=n i++) {
     for(j=0; j<=32; j++)
       cout << ch;
     cout << endl;
  }
}
```

actual parameters

function name

formal parameters

- Local variables

  - Variables declared inside a function are called *local variables*

  - These variables are only known to *main()* (they are only known to the function)

  - Local variables occupy memory space only when the function is called

  - In general, variables declared inside a block are called *local variables*

  - These variables are valid only inside the block they have been declared

  - Variables declared outside any function are called *global* variables

  - Global variables are known to any function !!