
```
double  multiplyTwo(int var, bool flag)  {
```

- `double` — *return* or *output parameter* type
- `multiplyTwo` — function name
- `(int var, bool flag)` — input parameters identifiers and types

body of the function:

```
  double product = 0;
  if (flag) {
    product = 2.0 * var;
  } else {
    product = var;
  }
  return product;
}
```

```
// This program prints a pattern of stars using two functions
#include <stdio.h>
// Function prototypes
void printPattern(int numOfRows);
void printStars(int numOfStars);

int main(void) {
  int lines;
  printf("Enter the number of lines in the pattern: ");
  scanf("%d", &lines);
  printPattern(lines);
  return 0;
}

void printStars(int numOfStars) {
  for (int star = 1; star <= numOfStars; star++) {
    printf("%c", '*');
  }
  printf("\n");  // print an endline to start a new line
}

void printPattern(int numOfRows) {
  for (int row = 1; row <= numOfRows; row++) {
    printStars(row);
  }
}
```

1. *return* or *output parameter* type
2. function *name*
3. *type* of *input* paramter
4. *input* paramter
5. body of the function

```
① ② ③ ④
void printStars(int numOfStars) {
  for (int star = 1; star <= numOfStars; star++) {
    printf("%c", '*');
  }

  printf("\n");  // to start a newline
}
```

```
void printPattern(int numOfRows) {

  for (int row = 1; row <= numOfRows; row++) {
    printStars(row);
  }
}
```

Call `printStars` → `printStars(row);`

Pass `row` **value** to `numOfStars` in `printStars` function

Function Prototype

```
void printStars(int numOfStars);
```

Also a Function Prototype

```
void printStars(int);
```

not required

```
// Function prototype
<return type> functionName(<type>);

int main(void) {
  // Call function
  <type> variableName = functionName(<variable to pass>);

  return 0;

}
// Function implementation
<return type> functionName(<type> <input parameter name>) {

  return <variable with same type as <return type>>;
}
```

```
// This program prints a pattern of stars using 2 functions
#include <stdio.h>

// Function prototypes: Headers of functions
void printPattern(int numOfRows);
void printStars(int numOfStars);
```

1. every program starts with executing main
2. `printPattern` is called
3. value of `lines` is passed to `numOfRows`
4. `printStars` is called
5. value of `row` is passed to `numOfStars`
6. body of `printStars` is executed
7. returns to where `printStars` was called

4 to 7 repeats depending on `numOfRows`

8. returns to where `printPattern` was called

```
① int main(void) {
    int lines;
    printf("Enter the number of lines in the pattern: ");
    scanf("%d", &lines);
② printPattern(lines);
    return 0;
  ⑧ }

③ void printPattern(int numOfRows) {
    for (int row = 1; row <= numOfRows; row++) {
  ⑦ ④ printStars(row);
    }
  }

⑤ void printStars(int numOfStars) {
    for (int star = 1; star <= numOfStars; star++) {
  ⑥   printf("%c", '*');
    }
    printf("\n");  // to start a newline
}
```