CMSC21
FUNDAMENTALS OF PROGRAMMING

Kristine Bernadette P. Pelaez

Institute of Computer Science
University of the Philippines Los Baños

FUNCTIONS

FUNCTIONS

A block of code that
performs a certain task.

FUNCTIONS

enforces modularity

FUNCTIONS

enforces **modularity**
increases **code reusability**

FUNCTIONS

enforces **modularity**
increases **code reusability**
easier **code management**

FUNCTION DEFINITION

```
returntype fncName([type var,…]){
    //statement_1
    //statement_2
    //…
    //statement_N
}
```

FUNCTION DEFINITION

```
int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

```
int square(int a){
  return a*a;
}
```

FUNCTION CALL

Invokes the function
or **starts the execution of the function**.

FUNCTION CALL

```c
int main(){

  return 0;
}
```

```c
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FUNCTION CALL

```
int main(){
  getInteger();
  return 0;
}
```

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FUNCTION CALL

```
int main(){
  int a;
  a = getInteger();
  return 0;
}
```

The variable a will hold the value that will be returned by the `getInteger()` function.

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FUNCTION CALL

```
int main(){
  int a;
  a = getInteger();
  square(a);
  return 0;
}
```

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FUNCTION CALL

```
int main(){
  int a,ans;
  a = getInteger();
  ans = square(a);
  return 0;
}
```

The variable **ans** will hold the value that will be returned by the **squared()** function.

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FUNCTION PARAMETERS

Two types

ACTUAL

FORMAL

FUNCTION PARAMETERS

ACTUAL

can be found on the
function call

FUNCTION CALL

```
int main(){
  int a,ans;
  a = getInteger();
  ans = square(a);
  return 0;
}
```

ACTUAL

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FUNCTION PARAMETERS

FORMAL

can be found on the **function definition**

FUNCTION CALL

```
int main(){
  int a,ans;
  a = getInteger();
  ans = square(a);
  return 0;
}
```

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FORMAL

FUNCTION CALL

```
int main(){
  int a,ans;
  a = getInteger();
  ans = square(a);
  return 0;
}
```

```
int square(int a){
  return a*a;
}

int getInteger(){
  int num;
  printf("int: ");
  scanf("%d",&num);
  return num;
}
```

FORMAL

EXAMPLE!

LET'S TRY!

FUNCTION PROTOTYPES

Also known as

Function declaration

FUNCTION PROTOTYPES

Allows you to call a function that is defined **after the function call**.

FUNCTION PROTOTYPES

They are usually placed **after the preprocessor directives**.

NEXT MEETING

RECURSION

CMSC21
FUNDAMENTALS OF PROGRAMMING

Kristine Bernadette P. Pelaez

Institute of Computer Science
University of the Philippines Los Baños