CS11600: Introduction to Computer Programming (C++)

Lecture 6

Svetlozar Nestorov
University of Chicago

Outline

- Functions
- Passing arguments
- Return values
- Side effects
- Inline functions
- Recursion

Functions

- Group of statements that implement some task.
- Why use functions?
  - Design
  - Repetition
  - Modularity
  - Debugging

Declarations and Definitions

- Function declaration:

```
Type name(Type1 arg1, Type2 arg2);
```

type of return value; name of function; type of argument; name of argument

- Argument names are optional.
- Function definition:

```
Type name(Type1 arg1, Type2 arg2)
{ statements; }
```

- A return type `void` means that the function does not return a value.

Forward Referencing

- A function must be declared or defined before it is called.
- Problem for mutually recursive functions.

```
void egg();
void chicken() {
  egg();
  …
}
void egg() {
  chicken();
  …
}
```


Arguments

- Values of arguments are allocated on the stack.
- Arguments are passed by *value* or *pointer*.
- Arrays are always passed by pointer!

```
void f(int x, int *y) {
   x++, (*y)++;
}
int main() {
  int x = 10, y = 10;
  f(x,&y);
    cout << x << y;
}
```

`&name` is the memory address of name.

Return Values

- Basic form:

  `return expr;` or
  `return;` if return type is `void`

- Returning pointers or references.

```
int* g(int x) {
    int y = 10;
    if (x > y) {
        return &x;
    }
    return &y;
}
```

Bad idea!
But compiler only gives a warning.

Side Effects

- Functions may modify arguments passed by pointers.
- Functions should return pointers only to dynamically allocated memory.

Inline Functions

```
inline Type name(Type1 arg1);
```

- Keep logical organization without the overhead of a function call.
- Compiler substitutes a call to an inline function with code implementing the function.
- Improves performance at the expense of program size.
- Mainly used for very short functions (one-liners).
- Better than macros (`#define`).

Recursive Functions

- Simplify programming at the expense of performance (stack push and pop overhead).
- However, iteration may be faster.
- Example: Write a function that computes the greatest common divisor of two integers.

Main

- Every C++ program has a main function.
- Run program means *call main*.

```
int main (int argc, char *argv[])
{ statements; }
```

- `argc` is the number of arguments.
  - At least 1 (program name)
- `argv` is the array of arguments as `char*`
- `argv[0]` is the name of the program.