CS 131

Basic Expressions and Functions

Plan for today

- Initial types
- Three important features of expressions
- Examples of basic expressions
- Making functions

TYPES: What data *can be*

- A **type** determines what values something can have
- Two for now: **int** (integer) and **char** (character)
- int: Whole numbers. values like 0,1,2,3,-4,-79, 42
- char: Single screen-displayable symbol. values like ‘a’, ‘W’, ‘w’, ‘!’, ‘6’, and ‘ ’ (a space)

About Expressions

- Your programs are a sequence of expressions
- Expressions have four important features:
  - VALUE : “What it *means*”
  - SIDE EFFECTS : “What it *does*”
  - TYPE: “What values can *can it be*"
  - SEMANTICS: “What *kind* of expression is it?
- As you encounter new expressions and work with old ones, practice asking “What does this *mean*? What does this *do*? What *can it be*? What *kind* of expression is it?”

Example Expressions

- printf(“Hello, World”)
- “Hello, World”
- printf(“Hello %s World”, “something funny”)
- 5
- printf(“I have %d fingers on my hand”, 5)
- return 0

printf(“Hello, World”)

- DOES: Puts the first (and only) argument on screen
- MEANS: 12 (Number of letters put on screen)
- TYPE: **`int`**eger
- KIND: function call

“Hello, World”

- DOES: Nothing
- MEANS: The string of 12 letters: `Hello, World`
- TYPE: **`char*`** (a “C-style string”)
- KIND: literal, string

printf(“Hello %s World”, “something funny”)

- DOES: Substitutes the second argument into the place of %s in the first quoted text, and then puts the result on screen.
- MEANS: 27 (Number of letters put on screen)
- TYPE: **`int`**eger
- KIND: function call

5

- DOES: Nothing
- MEANS: The number 5
- TYPE: **`int`**eger
- KIND: literal, integer

printf("I have %d fingers on my hand", 5)

- DOES: Substitutes the second argument (5) into the place of %d in the first argument (I have %d fingers on my hand), and then puts the modified result on screen.
- MEANS: 27 (Number of letters put on screen)
- TYPE: **`int`**eger
- KIND: function call

printf("I have %d fingers on my hands", 10)

- DOES: Substitutes the second argument (10) into the place of %d in the first argument (I have %d fingers on my hand),
  and then puts the modified result on screen.
- MEANS: 29 (Number of letters put on screen)
- TYPE: **`int`**eger
- KIND: function call

return 0

- DOES: Ends the current function immediately and gives the function call the value 0
- MEANS: This expression has no meaning
- TYPE: This expression has no type
- KIND: function return

Function basics

- Functions are a named collection of expressions
- Using a function needs a function call expression
- The *meaning* of a function call is “The meaning of each expression in the function, one after another”
- The *value* of a function call is “The value that accompanies the first return expression seen”
- The *type* of a function call is “The function’s type”

Program with function

```c
    /** @file 120831_prog_with_function.c
 *  @brief happyhappyhappybirthday!
 *  @author Prof. Adams
 */

#include <stdio.h>

/**
 @brief Give someone a happy birthday.
 @param toWhom The name of the person you're happy for
 @param theirAge Hold old they are
 @return How many candles to put on their cake

 Says happy Nth birthday a few times, with the third time
 being directed at the person.
*/
int happyBirthdayTo(char* toWhom, int theirAge);

int main()
{
    happyBirthdayTo("pet rock", 19);
    happyBirthdayTo("Richard Gere", 63);
    return 0;
}

int happyBirthdayTo(char* toWhom, int theirAge)
{
    printf("Happy %dth birthday to you, ", theirAge);
    printf("happy %dth birthday to you, ", theirAge);
    printf("happy %dth birthday dear %s... ",
           theirAge, toWhom);
    printf("happy %dth birthday to you!\n", theirAge);
    return theirAge+1;
}
```

Parts of a function

- Documentation
- Declaration
- Definition
- Calls

Function Declaration

- Comes before ‘int main(...)’.
- Consists of HEADER followed by SEMICOLON
- Has type of function, name of function, parens containing optional comma-separated list of argument declarations, and a semicolon

- int averageOf(int x, int y);
- char someLetter();
- `int happyBirthdayTo(char* toWhom, int theirAge);`

Function Definition

- Placed after the body of ‘int main(...)’
- Consists of a HEADER and a BODY
- Header
  - Just like in declaration (but NO SEMICOLON)
- Body
  - A bunch of statements (1 per line)
  - Enclosed in {braces}
  - Needs a return statement

Definition Example

```
int happyBirthdayTo(char* toWhom, int theirAge)
{
  printf("Happy %dth birthday to you, ", theirAge);
  printf("happy %dth birthday to you, ", theirAge);
  printf("happy %dth birthday dear %s... ", theirAge, toWhom);
  printf("happy %dth birthday to you!\n", theirAge);

  return theirAge+1;
}
```

Function Calls

- An expression
  - DOES: Stuff in the function body
  - MEANS: Whatever is returned
  - CAN BE: The declared return type
  - KIND: Function call
- Syntax: functionName(arguments)
- Example:

```
happyBirthdayTo("pet rock", 19)
```

Function Documentation

- Position: Right above the declaration
- Appearance: Surrounded by `/** Doxygen Comment Marks */` (notice the **two** lead stars)
- Content: Brief description.
  Long description.
  @param information
  @return information
  other notes

Documentation Example

```
 /**
Give someone a happy birthday.

Says happy Nth birthday a few times, with the third
time being directed at the person.

@param toWhom The name of the person you're happy for
@param theirAge Hold old they are
@return How many candles to put on their cake
*/
int happyBirthdayTo(char* toWhom, int theirAge);
```