임베디드컴퓨팅
Embedded Computing
(0009488)
C/C++ in Arduino, part2

2022년 2학기

정보기술대학 정보통신공학과
김 영 필
ypkim@inu.ac.kr

인천대학교
INCHEON NATIONAL UNIVERSITY

Reminder: what we knew about C

- Comments
- Numeric constant
- Basic datatypes (int, float, char, String …)
- Naming rules for Identifiers
- Reserved names - Keywords
- Arithmetic expressions and operators

How can we

Arduino C/C++ Language Structural view

Structure of C Source files

- A source program can be divided into one or more "source files," or "translation units."
  - The input to the compiler is called a "translation unit."

- ***translation-unit:***

  external-declaration

  translation-unit external-declaration

- ***external-declaration:***

  function-definition

  declaration

Ref. - https://docs.microsoft.com/en-us/cpp/c-language/source-files-and-source-programs?view=msvc-160

declaration

- declaration-specifiers init-declarator-list$_{opt}$;

  is essential!

- declaration-specifiers:
  storage-class-specifier declaration-specifiers$_{opt}$
  type-specifier declaration-specifiers$_{opt}$
  type-qualifier declaration-specifiers$_{opt}$

  opt == optional

- init-declarator-list:
  init-declarator
  init-declarator-list , init-declarator

  is used for expressing lists!

- init-declarator:
  declarator
  declarator = initializer

  is used for initial assignment of variable

storage-class-specifier

- The "storage class" of a variable determines
  - whether the item has a [redacted] lifetime.
  - C calls these two lifetimes **"static"** and **"automatic."**
    - An item with a global lifetime **exists and has a value throughout the execution of the program.**
  - All **functions** have **global** lifetimes.
- Syntax
  - storage-class-specifier:

    auto
    register
    static
    extern
    typedef

[redacted]

[redacted]

construct shorter or more meaningful names for types already defined by C or for types that you have declared.

Ref. - https://docs.microsoft.com/en-us/cpp/c-language/c-storage-classes?view=msvc-160

declaration

- declaration-specifiers init-declarator-list$_{opt}$;

- declaration-specifiers:
  storage-class-specifier declaration-specifiers$_{opt}$
  type-specifier declaration-specifiers$_{opt}$
  type-qualifier declaration-specifiers$_{opt}$

- init-declarator-list:
  init-declarator
  init-declarator-list , init-declarator

- init-declarator:
  declarator
  declarator = initializer

type-specifiers

- Define the type of a variable or function declaration.
- Syntax
  - type-specifier:

    void
    char
    short
    int
    long
    float
    double
    signed
    unsigned
    struct-or-union-specifier
    enum-specifier
    typedef-name

Ref. - https://docs.microsoft.com/en-us/cpp/c-language/c-type-specifiers?view=msvc-160

declaration

- declaration-specifiers init-declarator-list$_{opt}$;

- declaration-specifiers:
  storage-class-specifier declaration-specifiers$_{opt}$
  type-specifier declaration-specifiers$_{opt}$
  type-qualifier declaration-specifiers$_{opt}$

- init-declarator-list:
  init-declarator
  init-declarator-list , init-declarator

- init-declarator:
  declarator
  declarator = initializer

type-qualifier

- Give one of two properties to an identifier.
  - **const**
    - declares an object to be
  - **volatile**
    - declares an item whose value can legitimately be changed by something beyond the control of the program (e.g. concurrently executing thread)
  - **restrict**
    - gives an optimization hint to the compiler that no other pointer in the current scope refers to the same memory location
- Syntax
  - type-qualifier:

    const

    restrict

    volatile

Ref. - https://docs.microsoft.com/en-us/cpp/c-language/type-qualifiers?view=msvc-160

declaration

- declaration-specifiers init-declarator-list$_{opt}$;

- declaration-specifiers:
  storage-class-specifier declaration-specifiers$_{opt}$
  type-specifier declaration-specifiers$_{opt}$
  type-qualifier declaration-specifiers$_{opt}$

- init-declarator-list:
  init-declarator
  init-declarator-list , init-declarator

- init-declarator:
  declarator
  declarator = initializer

declarator

- The part of a declaration that specifies
  - the name to introduce into the program. It can include modifiers * (pointer-to)
- Syntax
  - declarator:

    pointer_{opt} direct-declarator
  - direct-declarator:

    identifier

    (declarator)

    direct-declarator [$\text{constant-expression}_{opt}$]

    direct-declarator (parameter-type-list)

    direct-declarator ($\text{identifier-list}_{opt}$)

    function declarator. explained later.
  - pointer:

    * $\text{type-qualifier-list}_{opt}$

    * $\text{type-qualifier-list}_{opt}$ pointer
  - type-qualifier-list:

    type-qualifier

    type-qualifier-list type-qualifier

declaration

- declaration-specifiers init-declarator-list$_{opt}$;

- declaration-specifiers:
  storage-class-specifier declaration-specifiers$_{opt}$
  type-specifier declaration-specifiers$_{opt}$
  type-qualifier declaration-specifiers$_{opt}$

- init-declarator-list:
  init-declarator
  init-declarator-list , init-declarator

- init-declarator:
  declarator
  declarator = initializer

initializer

- initializer:

  assignment-expression

  { initializer-list }

  { initializer-list , }

- initializer-list:

  $\text{designation}_{\text{opt}}$ initializer

  initializer-list , $\text{designation}_{\text{opt}}$ initializer

assignment-expression

- expression:

  assignment-expression

  expression , assignment-expressi

- assignment-expression:

  conditional-expression

  unary-expression assignment-operator assignment-expression

- conditional-expression:

  logical-OR-expression

  logical-OR-expression ? expression : conditional-expression

initializer

- initializer:

  assignment-expression

  { initializer-list }

  { initializer-list , }

- initializer-list:

  $\boxed{\text{designation}_{opt}}$ initializer

  initializer-list , designation_{opt} initializer

designation

- designation:

  designator-list =

- designator-list:

  designator

  designator-list designator

- designator:

  [constant-expression]

  . identifier

conditional-expression

accessing to

accessing to structure member variables (C/C++) or functions (C++)

Structure of C Source files, again

- A source program can be divided into one or more "source files," or "translation units."
  - The input to the compiler is called a "translation unit."

- ***translation-unit:***

  external-declaration

  translation-unit external-declaration

- ***external-declaration:***

  function-definition

  declaration

Ref. - https://docs.microsoft.com/en-us/cpp/c-language/source-files-and-source-programs?view=msvc-160

function-definition

- specifies the name of the function, the types and number of parameters it expects to receive, and its return type.
- also includes a function body with the declarations of its local variables, and the statements (what the function does)
- Syntax
  - function-definition:

    declaration-specifiers$_{opt}$ declarator declaration-list$_{opt}$ compound-statement
  - declarator:

    pointer$_{opt}$ direct-declarator
  - direct-declarator:

    direct-declarator (parameter-type-list)

    direct-declarator (identifier-list$_{opt}$)

function declarator

Ref. - https://docs.microsoft.com/en-us/cpp/c-language/c-function-definitions?view=msvc-160

compound statement

- describes function body (one or more statements)
  - delimited by braces ({ })
- syntax
  - compound-statement:

$$\{\ \text{declaration-list}_{opt}\ \text{statement-list}_{opt}\ \}$$

Variables declared in a function body, known as **local variables**

Execution control passes to the [redacted] in compound-statement and continues until a [redacted] is executed or **the end of the function body** is encountered.

Control then **returns to** the point **at which the function was called**.

Concept of statement

- Control the flow of program execution.
- C statements are available
  - to perform loops
  - to select other statements to be executed
  - to transfer control

- Again, syntax of statement-list
  - statement-list:

    statement

    statement-list statement

statement

- Syntax
  - statement:

    compound-statement

    labeled-statement

    jump-statement

    expression-statement

    selection-statement

    iteration-statement

jump-statement:
  goto identifier **;**

labeled-statement:
  identifier **:** statement

expression-statement:
  $\text{expression}_{\text{opt}}$ **;**

selection statement

- controls
  - The body of an if statement is executed if the value of the expression is nonzero.
- syntax
  - selection-statement:

    if (expression) statement

    if (expression) statement **else** statement

statement

- Syntax
  - statement:

    compound-statement

    labeled-statement

    jump-statement

    expression-statement

    selection-statement

    iteration-statement

iteration-statement

- lets you repeat a statement or compound statement until a
- syntax
  - iteration-statement:

    do statement **while** **(** expression **)** **;**

Again, Control Structure Keywords

- **break**
- **continue**
- do...while
- else
- **for**
- goto
- if
- return
- **switch...case**
- **while**

Assignment: Angry thermometer

- Write sketch program according to the requirements
- Requirements
  - Initial temperature (temp.)
    - Zero Celsius degree (C)
  - **(+)** Every 1000 ms, +3 C
    - When it exceeds 100 C
      - Stop increasing temp., and blink LEDs every 50 ms
    - After blinking 10 times
      - Stop blinking LEDs, and decrease the temp. 3 C every 1000 ms
    - When it comes 0 or less temperature,
      - Set it zero, and repeat **(+)**