C++ Features and Data Types

프로그래밍 입문(2)

Topics

- C++ 프로그램의 기초적인 문법 및 형태
    - main()함수, 전처리 지시자, iostream, 문장(statement), 코드 블럭(code block), 주석문(comment) 등
- **C++의 기본 자료형 (Primitive Data Types)**
    - **정수(integer), 실수(floating point), 문자(character), 불리언(boolean), void**
- C++의 다양한 연산자 (Operators)
    - 대입(assignment), 산술(arithmetic), 관계/비교(relation/comparison), 논리(logic), 비트(bitwise), 복합 대입(compound assignment) 연산자 등.

C++ Data Types

Variable

- 변수(Variable)는 값을 저장하기 위한 공간을 마련하고 이름을 붙인 것.

- 프로그램이 진행됨에 따라 변수의 값은 계속해서 변할 수 있기 때문에 변수라는 이름이 붙었습니다.

- 반대로 상수(constant)라는 한 번 값이 정해지면 변하지 않는 경우도 존재.

- 그 외 고정된 값을 표현하기 위한 literal이 있습니다.

Variable Declaration

- 변수를 사용하기 위해서는 우선 변수를 선언해야 합니다.
- 어떤 데이터형의 이름이 무엇인 변수를 사용하겠다고 선언.
- 기본적인 형태는 <변수의 데이터형> <변수 이름>;
  - e.g.) int a; double b; char c;
- <변수 이름>은 식별자(identifier)라고도 부릅니다.
- 주의!) C++에서는 변수를 선언만하면 변수에 garbage value가 들어가 초기값이 무엇이 될 지 예측할 수 없습니다.

Variable Initialization

- 따라서 변수를 선언할 때는 초기값을 지정해 주는 것이 좋습니다.
- 이것을 변수 초기화(initialization)이라고 합니다.
- 선언 시 <변수의 데이터형> <변수 이름> = <초기값> 형태로 값을 지정합니다.
- 초기값은 변수의 데이터형과 반드시 호환이 되어야 합니다.
- 초기값으로는 복잡한 표현식을 사용할 수도 있습니다.
  - `int a = b * (x - y) + 3;`

Variable Names

- 반드시 알파벳이나 언더스코어(_)로 시작.
- 첫 문자 이후는 문자나 숫자.
- Case Sensitive: `minValue`와 `minvalue`가 구분됨.
- 공백/특수문자 사용불가 (예외 '_').
- 예약어(keyword)는 사용 불가: return, int, void 등등.
- 가독성을 위해 간결하고 의미 있는 이름을 쓰는 것이 권장됨.

Variable Declaration Statement

- 변수 선언문에서는 한 번에 **동일한 데이터형**의 여러 변수를 선언하고 초기화하는 것도 가능합니다.
- 각 변수는 쉼표(,)로 구분합니다.
  - `int a, b = 3, c, d = 4;` → **OK**
  - `int x = 3, double y = 4.0;` → **NO**
  - `int x = 3; double y = 4.0;` → **OK**

Assignment Statement

- 이미 선언된 변수의 값은 대입문(assignment statement)으로 변경할 수 있습니다.
- 형태는 <변수 이름> = <값의 표현식>.
  - `a = 1; a = a + 1;` → **OK**
  - `x = y - i;` → **OK**
  - `x + i = y` → **NO!!**
- 왼쪽에는 Lvalue가 와야합니다.
- Lvalue: 접근 가능한 메모리 주소가 있어 값을 넣을 수 있는 것.
- x + i는 표현식으로 두 변수의 합을 의미할 뿐 특정 메모리에 접근하도록 지정된 것이 아님.

Constant

- 상수(constant)는 한 번 초기화하면 값을 변경할 수 없는 변수.
- 변경될 일이 없으며 코드 전반에 걸쳐 사용할 값을 상수로 정의하면 좋습니다.
- 선언을 위해서는 C처럼 `#define`이나 새로운 `const` 키워드를 사용.
- 상수 이름은 모두 대문자로 하는 것이 보통.
- #define DELIM ‘,’

const

- 형식: const <데이터형> <상수 이름> = <상수 값>
  - const char DELIM = ‘,’;
- 기존 변수 선언 앞에 const 키워드만 붙은 형태.
- #define에 비해 데이터형을 지정할 수 있다는 장점이 있습니다.

Constant vs. Literal

- Literal은 단순히 값만을 나타내는 것.
  - number literal: 0, -1, 3.5, character literal: ‘a’, ‘:’, ‘\n’
  - boolean literal: true, false
- 상수(Constant)는 값을 가리키는 식별자(Identifier)로 값을 표현.
  - MAX_SIZE = 256, DELIM = ‘,’

Constant vs. Literal

- 만약 코드 전반에 걸쳐 반복적으로 사용되는 값을 Literal로 사용한다면?
  - 그 값을 변경할 필요가 있을 때, 모든 위치를 찾아 변경해주어야 함.
  - 값 자체의 의미를 알기 어려워 가독성이 떨어집니다.
- 상수를 사용하면?
  - 상수가 정의된 위치 하나만을 변경해주면 됩니다.
  - 상수 이름으로 그 위치에 쓰인 값의 의미를 추측해 가독성이 좋아짐.
  - e.g.) csv파일을 읽어 데이터를 처리하는 프로그램에서,
    - `const char DELIM = ‘,’ → const char DELIM = ‘\t’`

Data Types

- Primitive Data Types: 또는 fundamental data types.
  - 가장 기본적인 데이터형으로 다른 데이터형을 위한 재료로도 사용됨.

- Compound Data Types: 또는 composite data types.
  - 배열(Array), 포인터(point), 구조체(struct), 클래스(class), etc.
  - Primitive type을 바탕으로 만들어 짐 - e.g., 배열 int[], char[]

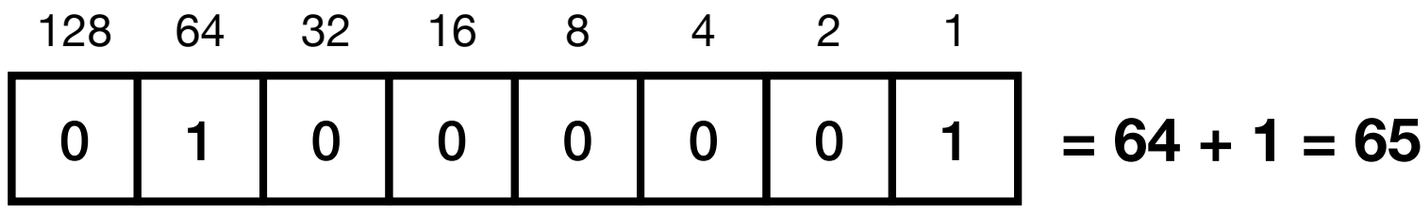
Primitive Types

- 크게 구분하면 다음의 종류가 있음.
  - Boolean type: true or false
  - Integer type: 정수형(0,1,…), 크기에 따라 short, int, long에 signed, unsigned가 붙음.
  - Character type: 문자형, 한 글자, ’a’, ‘b’ 와 같은 alphabet이나 ‘:’, ‘+’와 같은 기호 등을 포함.
  - Floating point type: 실수형, (0.101, 3.14,…), float, double 등.
  - Void type: empty.

Primitive Types

- 왜 다양한 데이터형이 존재하는가?
- 현실에 존재하는 다양한 데이터를 표현할 필요가 있기 때문.
- 실제로 메모리에 기록되는 것은 01001010....
- 이를 읽어왔을 때, 어떤 의미로 받아들일 것인가 하는 문제.

Primitive Types



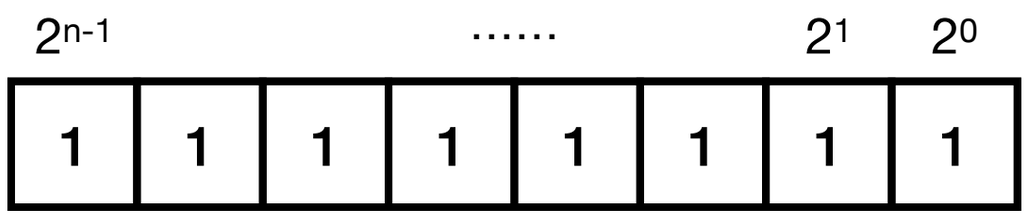
- 정수형으로 읽으면 65.
- Boolean으로 읽으면 true.
- 문자형(character)으로 읽으면 ‘A’ (ASCII 코드).
- 같은 값이 데이터형에 따라 의미가 달라집니다.

Integer Type

- 정수를 표현하기 위한 다양한 데이터 형이 존재.
- 각 정수 데이터형의 크기는 컴파일러나 시스템에 의존적이므로, C++에서는 최소 크기를 보장.
- `sizeof(<type>)`으로 크기를 확인할 수 있음.
  - `sizeof(int), sizeof(long)`
- 또는 `climits`에서 상수를 이용하여 확인.

데이터형	최소 크기 (bytes/bits)	일반적인 크기 (bytes/bits)
`char`	1 / 8	1 / 8
`short`	2 / 16	2 / 16
`int`	2 / 16	4 / 32
`long`	4 / 32	4 / 32
`long long`	8 / 64	8 /64

Range of Integer Type



int: 32 bits → $-2^{31} \sim 2^{31}-1$

- 각각의 정수형에는 signed와 unsigned가 존재.
- 음수는 2의 보수(2’s complement)로 계산합니다.
- 범위 계산하는 방법
  - unsigned: 비트 수만큼 숫자를 표현할 수 있으므로, n비트로 $0 \sim 2^n - 1$까지.
  - signed: 같은 범위를 양수/음수로 나누어 쓰므로, $-2^{n-1} \sim 2^{n-1}-1$까지 (0은 양수쪽으로 포함).
- 표현가능한 범위를 고려하여 정수 데이터형을 고르는 것이 좋습니다.

Character Type

- 한 개의 문자를 표현하는 데이터형으로 1 byte.
  - `char c = ‘x’;`
- 정수값에 대응하는 문자표(ASCII, EBCDIC 등)의 문자를 나타냄.
- 홑따옴표(‘’) 사이에 문자를 입력하는 식으로 값을 나타낼 수 있음.

Floating Point Type

- 부동소수점(浮動小數點)으로 실수를 나타냅니다.
- `float`, `double`, `long double` 세 가지가 있습니다.
- 정확히 정해진 크기는 없으며, `float`은 적어도 4 bytes, `double`은 `float`보다 작지 않으며 적어도 6 bytes, `long double`은 최소한 double과 같은 크기.
- 보통 `float`는 4 bytes, `double`은 8 bytes가 됨.

Floating Point Numbers

- 실수 자체를 저장하지 않고 실수에 대한 정보를 저장하는 방식.
- 실수의 표현형태는 3.14처럼 직접 표시하거나, 자리수가 많은 경우 9.11e-31처럼 지수표기로 표현합니다.
- 지수표기 형식: +1.12E+32
- 정밀도(precision) 문제
  - `cfloat`에서 `FLT_DIG, DBL_DIG, LDBL_DIG`로 정밀도 확인
  - `float`는 6자리, `double`은 15자리, `long double`은 18자리. 시스템별로 상이할 수 있음.
  - `float`는 유효숫자 6자리까지만 정확히 표현이 가능하다는 의미.

Precision

- `float`가 6자리까지만 유효숫자를 표현할 수 있다는 것의 의미.
  - `float a = 1.23E+20f; //1230....0`
  - `float b = 1.23E+6f; //1,230,000`
- a는 123,000,000,000,000,000,000, b는 1,230,000
- 여기에 1을 더한다면?
- b의 경우는 1.230001 곱하기 10의6제곱(E+6)이 됩니다.
- a의 경우는 1.2300…..01 곱하기 10의 20제곱이 됩니다.

Precision

- `float c = a + 1.0f;`
- `float d = b + 1.0f;`
- 그렇다면 다음의 값은 어떻게 표현이 되는가?
  - `c - a = ?`
  - `d - b = ?`
- 수학적으로는 당연히 둘 모두 1이지만, C++에서는 이것이 보장되지 않음.

Boolean Type

- 참과 거짓 두 가지 값을 표현하는 1 byte 데이터형.
- `true`, `false` 두 값을 가짐.
- 숫자 0은 `false`로, 그 외의 값은 `true`로 해석됨.
- `int`형과 호환 가능.
  - `bool x = -7; //x = true,  bool y = 0; //y = false`
  - `int a = true; //a = 1,  int b = false; //b = 0`

Void Type

- ‘없음’을 표시하기 위한 데이터형.
- 실제 변수의 데이터형을 나타내지는 않는 특수한 형태.
- 함수의 반환값이 없는 경우 등을 나타내기 위해 사용됩니다.

Summary

- 변수를 선언하는 방법
- 다양한 데이터형들
- 정수 데이터형의 표현가능한 숫자 범위
- 실수의 부동소수점 표현에서 정밀도 문제