제1장

C++ 기초

학습 목표

- C++ 기초 학습
  - C++ 기원, 객체지향 프로그래밍, 용어
- 변수, 식 및 할당문 학습
- 콘솔 입/출력 학습
- 프로그램 스타일 학습
- 라이브러리 및 네임스페이스 학습

C++ 기초

- C++ 언어의 기원
  - 저급 프로그래밍 언어
    - 기계어, 어셈블리
  - 고급 프로그래밍 언어
    - C, C++, ADA, COBOL, FORTRAN
  - 객체지향 프로그램 언어 C++
- C++ 용어
  - *프로그램* 및 함수 *functions*
  - cin and cout 객체를 사용한 데이터 입출력 수행

C++ 프로그램 예제 (1 of 2)

디스플레이 1.1 C++ 프로그램 예제

```
1  #include <iostream>
2  using namespace std;

3  int main( )
4  {
5      int numberOfLanguages;

6      cout << "Hello reader.\n"
7           << "Welcome to C++.\n";

8      cout << "How many programming languages have you used? ";
9      cin >> numberOfLanguages;

10     if (numberOfLanguages < 1)
11         cout << "Read the preface. You may prefer\n"
12              << "a more elementary book by the same author.\n";
13     else
14         cout << "Enjoy the book.\n";

15     return 0;
16 }
```

C++ 프로그램 예제 (2 of 2)

Sample Dialogue 1

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 0
Read the preface. You may prefer
a more elementary book by the same author.
```

← 사용자가 입력한 부분은 볼드체로 표현하였음

Sample Dialogue 2

```
Hello reader.
Welcome to C++.
How many programming languages have you used? 1
Enjoy the book
```

← 사용자가 입력한 부분은 볼드체로 표현하였음

C++ 변수

- C++ 식별자
  - 키워드/예약어 vs. 식별자
  - 대소문자 구분 및 합당한 식별자의 이름 정하기
  - 변수는 실제 의미를 전달하도록 이름을 정함
- 변수
  - 데이터를 저장하는 특정한 메모리의 공간을 의미함
  - 모든 변수는 사용되기 전에 반드시 먼저 선언되어야 함

데이터 형 설명

디스플레이 1.2 간단한 데이터형

데이터형	저장공간	값의 범위	정밀도
`short` (또는 `short int`)	2 bytes	–32,768에서 32,767까지	해당사항 없음
`int`	4 bytes	–2,147,483,648에서 2,147,483,647까지	해당사항 없음
`long` (또는 `long int`)	4 bytes	–2,147,483,648에서 2,147,483,647까지	해당사항 없음
`float`	4 bytes	대략 10^{-38}에서 10^{38}까지	7 digits
`double`	8 bytes	대략 10^{-308}에서 10^{308}까지	15 digits

데이터 형 설명

`long double`	10 bytes	대략 10^{-4932}에서 10^{4932}까지	19 digits
`char`	1 byte	모든 ASCII 문자들 (정수형으로도 사용될 수 있지 만 권장하는 바는 아니다.)	해당사항 없음
`bool`	1 byte	`true`, `false`	해당사항 없음

여기서 명시된 값들은 변수형에 따라서 서로 다른 값의 범위를 가진다는 것을 설명하기 위해서 몇몇 예시들로 나열되어 있다. 명시된 값들의 범위는 사용자의 컴퓨터 시스템에 따라 다를 수 있다. *정밀도*는 소수점 앞에 위치한 숫자를 포함해서 실제로 표현된 숫자의 개수를 의미한다. `float`, `double` 및 `long double`형은 양수 값의 범위를 가진다. 음수의 경우 각각 숫자 앞에 음수 기호가 표기된다는 사실만 다를 뿐, 비슷한 범위의 값을 가진다.

- byte 자료형 없음
- String을 기본 자료형으로 간주하지 않음(Java도 그러함)

변수 선언

- 변수를 선언하면서 초기화가 가능함
  - 변수의 값은 반드시 설정되어야 함!
    - 예: int myValue = 0;
- 프로그램 실행단계에서 변수의 값이 설정될 수 있음
  - Lvalues (좌측항 값) 및 Rvalues (우측항 값)
    - Lvalues 는 변수가 되어야 함
    - Rvalues 는 식으로 표현될 수 있음
    - 예:

      distance = rate * time;

      Lvalue: "distance"

      Rvalue: "rate * time"

데이터 할당문: 축약된 표현식

- 디스플레이 (14 p.)

예제	동일한 표현식
`count += 2;`	`count = count + 2;`
`total -= discount;`	`total = total - discount;`
`bonus *= 2;`	`bonus = bonus * 2;`
`time /= rushFactor;`	`time = time/rushFactor;`
`change %= 100;`	`change = change % 100;`
`amount *= cnt1 + cnt2;`	`amount = amount * (cnt1 + cnt2);`

백슬래쉬 문자상수

- 추가된 문자상수
- 백슬래쉬, \ 기호 다음에 오는 문자상수
  - 컴파일러는 백슬래쉬 다음에 오는 문자를 특별하게 해석함
  - 디스플레이 1.3에서 백슬래쉬 문자상수들의 종류를 보여줌

백슬래쉬 문자상수 종류 (1 of 2)

디스플레이 1.3 백슬래시 문자상수 종류

문자상수 종류	의미
\n	새로운 줄
\r	개행문자(현재 줄의 첫부분에 위치한 커서의 위치를 뜻하나 활용도는 많지 않음)
\t	(수평) 탭(다음 탭의 위치에 커서를 이동시킴)
\a	경고음(벨소리 등과 같은 경고음 발생)
\\	백슬래시(따옴표 안에 표현된 문장에서 백슬래시를 사용하게 함)
\'	따옴표(대부분 따옴표 안에 표현된 문장의 위치에 사용)
\"	이중따옴표(대부분 이중따옴표 안에서 표현된 문장의 위치에서 사용)
다음에 나열된 표현은 사용빈도수가 낮으나, 모든 경우를 고려하여 나열되었다.	
\v	수직 탭
\b	백스페이스
\f	폼 피드
\?	의문 부호

상수(const)

- Java의 final에 해당
- 상수 이름 짓기
  - 의미가 잘 전달되지 않는 문자상수를 사용하지 않는 것이 좋음
  - 상수 이름은 데이터의 의미를 실제로 표현할 수 있도록 정함
  - 예: const int NUMBER_OF_STUDENTS = 24;
    - " 상수 선언" 또는 " 상수 이름 짓기"
    - 상수는 프로그램 어느 부분에서도 사용이 가능함
    - const를 사용하여 상수의 값을 변경할 수 있음

상수 이름 짓기

디스플레이 1.4 이름이 부여된 상수

```
1  #include <iostream>
2  using namespace std;
3
4  int main( )
5  {
6      const double RATE = 6.9;
7      double deposit

8      cout << "Enter the amount of your deposit $";
9      cin >> deposit;
10     double newBalance;
11     newBalance = deposit + deposit*(RATE/100);
12     cout << "In one year, that deposit will grow to\n"
13          << "$" << newBalance << " an amount worth waiting for.\n";

14     return 0;
15 }
```

Sample Dialogue

```
Enter the amount of your deposit $100
```

산술 연산의 정밀도

- 산술 계산의 정밀도
  - 산술 연산의 정밀성은 매우 중요함!
    - 산술 연산의 정밀도 설정에 따라서 예상치 않은 결과를 얻을 수 있음!
  - 산술 연산 결과의 정밀도는 피연산자의 데이터 형에 따라 결정 됨
  - 산술 연산 결과의 정밀도 표현에 유의할 것!

산술 연산 결과의 정밀도를 보여주는 예제

- 예:
  - C++에서 17 / 5 산술 연산은 3으로 해석됨!
    - 2개의 피연산자들 모두 정수형
    - 따라서 정수의 나눗셈으로 해석 됨!
  - C++에서 17.0 / 5 연산은 3.4로 해석됨!
    - 피연산자 중에서 17.0은 double 형임
    - 따라서 나눗셈은 Double 형의 정밀도로 계산됨!
  - int intVar1 =1, intVar2=2;

    intVar1 / intVar2;
    - 정수형 나눗셈으로 계산됨!
    - 나눗셈 결과는 0임!

산술 연산 결과의 정밀도

- 산술 연산은 개별적으로 하나씩 평가함
  - 1 / 2 / 3.0 / 4 연산은 다음과 같이 3단계로 계산됨.
    - 첫 번째 연산→ 1 / 2 식은 0으로 계산 됨
    - 두 번째 연산→ 0 / 3.0 식은 0.0으로 계산 됨
    - 마지막으로→ 0.0 / 4 식은 0.0으로 계산됨!
- 따라서 산술 연산의 결과는 단순히 하나의 피연산자에 의하여 결정되지 않음
  - 결과적으로 산술 연산에 포함된 모든 피연산자들이 올바르게 실행될 수 있도록 유의해야 됨!
- 자꾸 말하지만 다음 코드의 결과는?
  - if (90 <= score <= 100)
    - score가 어떤 값이라도 항상 동일한 결과가 나옴!!

형 변환

- 두 가지 방법
  - 자동적 형 변환
    - 자동적으로 형 변환이 일어남

      예: 17 / 5.5

      여기서 17 정수는 자동적으로 17.0으로 형 변환이 이루어짐
    - 즉 17 → 17.0 으로 형 변환이 자동적으로 발생함
  - 강제 형 변환
    - 프로그래머가 명시적으로 형 변환 연산자를 사용하여 데이터 형을 변환시킴

      예: (double)17 / 5.5

      정수에 강제 형 변환 연산자를 사용함

      (double)myInt / myDouble

      변수에 강제 형 변환 연산자를 사용함

간단한 형태로 표현되는 연산자

- 증가 및 감소 연산자
  - 간략한 형식으로 표현된 연산자
  - 증가 연산자, ++

    intVar++; 표현은

    intVar = intVar + 1; 표현과 동일함
  - 감소 연산자, --

    intVar--; 표현은

    intVar = intVar – 1; 표현과 동일함

간단한 형태로 표현되는 연산자: 두 가지 유형

- 후위-증가 연산자: intVar++
  - 변수의 현재 값이 사용되고 이후로 값이 증가함
- 전위-증가 연산자: ++intVar
  - 변수의 값이 우선적으로 증가된 다음에 이 값이 사용됨

- 후위/전위 증가 연산자는 프로그램 상에서 서로 다르게 사용됨

- intVar++; 및 ++intVar; 와 같이 단독으로 사용되는 경우→ 결과는 동일함

후위-증가 연산자의 활용 예

- 프로그램 상에서 사용된 후위-증가 연산자 예:

```
int       n = 2,
          valueProduced;
valueProduced = 2 * (n++);
cout << valueProduced << endl;
cout << n << endl;
```

```
4
3
```

  - 후위-증가 연산자가 사용된 점에 유의할 것

전위-증가 연산자의 활용 예

- 프로그램 상에서 사용된 전위-증가 연산자 예:
  위의 코드가 실행되면 다음과 같은 값이 출력됨:

```
int       n = 2,
          valueProduced;
valueProduced = 2 * (++n);
cout << valueProduced << endl;
cout << n << endl;
```

```
6
3
```

  - 전위-증가 연산자가 사용된 점에 유의할 것

콘솔 입/출력

- 입/출력 객체 cin, cout, cerr
- <iostream> C++ library 에 정의되어 있음
- 프로그램 파일에서 처음으로 시작하는 부분에 명시하여야 하고 (전처리 지시자) 라고 호칭함:
  - #include <iostream>

    using namespace std;
  - 위와 같이 코드를 삽입해야 입/출력 객체인 cin, cout, cerr 를 사용할 수 있음

콘솔 입/출력

- 어떤 값들이 화면에 출력될 수 있나?
  - 어떤 유형의 데이터일지라도 컴퓨터 화면에 출력할 수 있음
    - 변수
    - 상수
    - 문자상수
    - 표현 식 (위에서 명시한 값들을 모두 포함할 수 있음)
  - cout << numberOfGames << " games played.";

    여기사 2 개의 값들이 출력됨:

    변수 numberOfGames의 값과,

    문자상수인 " games played.“ 이 화면에 출력됨
- 연속 출력: cout 객체 하나를 사용하여 여러 값들을 출력할 수 있음.

새로운 줄에서 출력하는 방법

- 새로운 줄에서 출력하기
  - 특수 문자상수인 "\n" 이 새로운 줄에서 출력할 때 사용됨
- 두 번째 방법은 endl 객체를 사용함
- 예:
  - cout << "Hello World\n";
    - "Hello World" 스트링에 "\n“ 문자상수를 첨가하면 스트링이 새로운 줄에서 출력됨
  - cout << "Hello World" << endl;
    - 위의 결과와 동일함

스트링 종류

- C++ 에서 여러 문자들로 조합된 “스트링” 데이터 형이 존재함
  - 기본적인 데이터 형이 아니며 나중에 자세히 설명 됨
  - 프로그램 시작부분에 **#include <string>** 이 삽입되어야 함
  - “+” 연산자를 두 개의 스트링에 사용하면 하나로 연결된 스트링이 됨
  - cin >> str 코드는 스트링을 읽는 과정에서 첫번째 공백문자를 만나면 작동이 중단됨

입/출력 예: (1 of 2)

디스플레이 1.5 cin과 cout를 이용한 스트링 사용법

```
1  //Program to demonstrate cin and cout with strings
2  #include <iostream>
3  #include <string>
4  using namespace std;
5  int main( )
6  {
7    string dogName;
8    int actualAge;
9    int humanAge;

10   cout << "How many years old is your dog?" << endl;
11   cin >> actualAge;
12   humanAge = actualAge * 7;

13   cout << "What is your dog's name?" << endl;
14   cin >> dogName;

15   cout << dogName << "'s age is approximately " <<
16         "equivalent to a " << humanAge << " year old human."
17         << endl;

18   return 0;
19 }
```

string 클래스를 사용하는 데 필요함

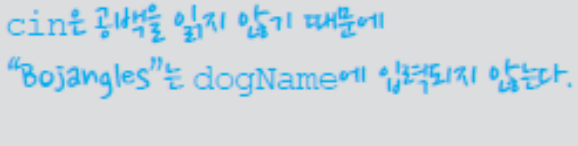
입/출력 예: (2 of 2)

Sample Dialogue 1

```
How many years old is your dog?
5
What is your dog's name?
Rex
Rex's age is approximately equivalent to a 35 year old human.
```

Sample Dialogue 2

```
How many years old is your dog?
10
What is your dog's name?
Mr. Bojangles
Mr.'s age is approximately equivalent to a 70 year old human.
```



출력 형식 정하기

- 숫자를 화면에 출력하는 형식 정하기
  - 변수의 값을 출력할 때 예상치 않은 형태로 출력 될 수 있음!

    cout << "The price is $" << price << endl;
    - 여기서 price (double 형) 변수의 값은 78.5이지만, 화면에는:
      - The price is $78.500000 또는:
      - The price is $78.5 으로 출력 될 수 있음
- 숫자를 화면에 보여줄 때 출력 형식을 명확히 정의해야 됨!

출력 형식 정하기

- 소수점 이하의 개수를 표현하는 데 사용되는 “마술공식“ :

  cout.setf(ios::fixed);

  cout.setf(ios::showpoint);

  cout.precision(2);

- 위의 문장들을 프로그램에서 한번 명시하면 프로그램의 나머지 부분에 모두 적용됨:
  - 소수점 이하 두 개의 숫자로 출력하려면
  - 예: cout << "The price is $" << price << endl;
    - 위의 문장이 실행되면 결과는 다음과 같음:

      The price is $78.50
- 필요하다면 출력되는 소수점 이하의 개수를 변경할 수 있음

출력 오류

- cerr 출력문
  - cerr 객체는 cout 객체와 동일한 원리로 작동됨
  - Cerr 객체를 사용하면 정상적인 출력과 오류가 발생하였을 때 출력을 구분할 수 있음.
- 컴퓨터 화면이 아닌 다른 장치에 출력하기
  - 대부분의 컴퓨터 시스템은 cout 및 cerr 객체를 사용하여 컴퓨터 화면이 아닌 다음과 같은 장치에 출력 할 수 있음:
    - 예: 프린터, 출력파일, 오류 콘솔, 등등.

cin 객체를 사용한 입력

- cin 객체는 입력, cout 객체는 출력을 담당함
- 차이점:
  - ">>" (입력 연산자) 를 사용하며 연산자의 방향이 데이터로 접근한다고 이해하면 좋음
  - 입력은 "cin" 객체를 사용함
  - cin 객체는 문자상수를 읽을 수 없으며 대부분의 경우 변수의 값을 읽어드림
- cin >> num;
  - 위의 문장이 실행되면 키보드에서 입력을 기다림
  - 사용자가 값을 입력하면 num 변수의 값으로 저장됨

cin and cout 객체 사용을 위한 문장구성

- 입력을 위해서 별도의 문장을 첨가할 것

  cout << "Enter number of dragons: ";

  cin >> numOfDragons;

  - 위에서 "\n" 문자상수를 사용하지 않았음. 위의 코드가 실행되면 다음과 같은 형태로 사용자가 카보드에 데이터를 입력하길 기다림:
    - Enter number of dragons: ____
    - 여기서 밑줄은 키보드를 통한 입력부분을 의미함
- 모든 cin 객체는 cout를 사용한 별도의 문장을 쓸 것을 권장
  - 이는 사용자가 입/출력을 편리하게 수행하고자 함

C언어에서의 입출력

- 출력 함수 : printf()

```
int i=7 ; double x =7.0 ; float y = 7.0 ;
printf(“%d%f%f\n”, i, x, y) ;
printf(“%.2f\n”, x) ;
printf(“%10.4f\n”, y) ;

char c = ‘X’ ;
printf(“%c\n”, c) ;   // putchar()

char str[16] = “Hello!” ;
printf(“%s\n”, str) ;
```

- 입력 함수 : scanf_s()

```
int i ; double x ; float y ;
scanf_s(“%d%lf%f”, &i, &x, &y) ;

char c ;
scanf_s(“%c”, &c) ;  // getchar()

char str[10] ;
scanf_s(“%s”, str, sizeof(str)) ;
```

프로그램 스타일

- 핵심: 프로그램을 쉽게 이해할 수 있고
  나중에 편리하게 수정될 수 있도록 구성되어야 함
- 주석을 표현하는 2 가지 방법:
  - // 2개의 슬래쉬를 연속으로 사용하면 같은 줄에 표현된 모든 문장은 주석으로 해석됨
  - /* 주석기호를 사용하면 */ 주석기호가 나올 때까지의 모든 문장은 주석으로 해석됨
  - 두 가지 방법 모두 널리 사용됨
- 식별자의 이름을 정하는 경우
  - 상수는 ALL_CAPS 예와 같이 대문자 위주로
  - 변수는 소문자와 대문자를 적절히 혼용
  - 가장 중요한 사항은 이름이 프로그래머에게 의미를 전달해야 됨

라이브러리

- C++ 표준 라이브러리
- #include <라이브러리 이름>
  - 위의 방식으로 사용자가 작성한 프로그램에 기존의 라이브러리를 첨가함
  - “전처리 지시자“ 라고 호칭함
    - 작성된 프로그램이 컴파일 되기 전에 먼저 실행되며 프로그램에 라이브러리 파일이 복사됨
- C++ 은 많은 라이브러리 파일들을 가지고 있음
  - 예: 입/출력, 산술, 스트링 라이브러리, 등등.

네임스페이스

- 네임스페이스 정의:
  - 네임(이름)들의 정의에 대한 집합
- 우리의 경우 "std" 네임스페이스에 초점 집중
  - 여기에 우리가 필요한 표준 라이브러리에 대한 정의가 나열되어 있음
- 예:

```
#include <iostream>
using namespace std;
```

  - 모든 표준 라이브러들의 이름들이 정의 되어있음
-

```
#include <iostream>
using std::cin;
using std::cout;
```

  - std를 사용하여 사용하고자 하는 객체를 활용할 수 있음

요약 1

- C++ 언어는 대소문자를 구별함
- 변수 및 상수의 이름을 정할 때는 실제적인 의미가 전달되도록 할 것
- 변수는 사용되기 전에 반드시 미리 선언되어야 하며 값이 초기화되어야 됨
- 숫자에 대한 산술연산을 수행할 때는 다음과 같은 사항들의 유의해야 됨
  - 정밀도, 괄호, 연산 순서
- 필요에 따라 #include 를 이용하여 C++ 라이브러리를 포함시킬 것

요약 2

- cout 객체
  - 콘솔 출력을 위해 사용함
- cin 객체
  - 콘솔 입력을 위해 사용함
- cerr 객체
  - 에러 메시지 출력을 위하여 사용함
- 프로그램의 가독력 향상을 위해서 주석을 사용함
  - 주석을 과도하게 사용하지 말 것