1장 C++ 기초

1.1 C++ 개요

특징

- C++ 언어는 C언어의 특성(저급 프로그래밍 언어+고급 프로그래밍)과 객체지향 언어의 특성을 가지고 있다.

- 객체지향 특성: 클래스(데이터+함수), 캡슐화(정보 은닉), 상속(코드 재사용), 다형성 등 …

- C++ ∋ C, C++  C

예제: Visual Studio 컴파일 & 실행 하기

ex\_00.cpp

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  cout << "Hello World, C++ " << endl;  return 0;  } |

예제: 구의 부피

ex01\_01.cpp

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  double radius, volume;  double PI = 3.14;  cout << "Enter a radius: ";  cin >> radius;  volume = (4.0 / 3) \* PI \* radius \* radius \* radius;  cout << "The volume is " << volume << endl;  return 0;  } |

- C++ 표준은 void main() 대신에 int main()를 쓴다.

- C++은 변수 선언이 중간에서도 가능하다.

1.2콘솔 입/출력

* cin, cout

|  |  |  |  |
| --- | --- | --- | --- |
| 입/출력 | C언어  #include <stdio.h>  int a, b; | C++언어  #include <iostream>  int a, b; | 종류 |
| 입력 | scanf(“%d %d”, &a, &b); | cin >> a >> b; | 값(int, char, float), 문자열 |
| 출력 | printf(“%d %d\n”, a, b); | cout << a << b << endl; | 변수, 값(int, char, float), 문자열, 표현식 |

- 연산자 방향(>>)으로 데이터가 흐른다고 이해 하라.

- “\n”과 endl은 동일하다.

- cin >> a >> b;와 cout << a << b << endl;에서 a와 b의 구분은 공백이다.

* 소수점 이하 숫자 표현

- 소수점 이하 두 개의 숫자 출력

|  |
| --- |
| cout.setf(ios::fixed); // 소수점을 고정시키는 표현  cout.setf(ios::showpoint); // 뒤에 붙는 소수점을 출력  cout.precision(2); // 소수점 2자리까지 출력 |

- 한번 명시하면 프로그램의 나머지 부분에 모두 적용된다.

예제: 구의 부피 – 두 개의 입력, 소수점 5자리 출력, “\n”

ex01\_02.cpp

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  double radius, volume;  cout.setf(ios::fixed);  cout.setf(ios::showpoint);  cout.precision(2);  cout << "Enter a radius and PI: ";  double PI;  cin >> radius >> PI;  volume = (4.0 / 3) \* PI \* radius \* radius \* radius;  cout << "The volume is " << volume << "\n";  return 0;  } |

1.3 라이브러리와 네임스페이스

* C와 C++의 라이브러이(헤드파일) 비교

- 역할: C/C++의 능력을 확장하고 이식성을 높이기 위함이다.

- 작성된 프로그램이 컴파일 되기 전에 먼저 실행되며 프로그램에 라이브러리 파일이 복사된다.

C 표준 라이브러리

- 예: #include <stdio.h> … <stdlib.h>, <string.h>

C++ 표준 라이브러리(STL-Standard Template Library)

- 확장자가 존재하지 않는다.

예: #include <iostream> … <string>, <vector>

- C 표준 라이브러리를 C++에서 사용하기 위해서 변환시켰다. “.h”를 제거하고 파일명 앞에 “c”를 붙인다.

예: #include <stdlib.h> → #include <cstdlib>

- 주의: <cstring>은 C의 <sting.h>이고, <string>은 C++의 STL로 둘 다 문자열을 처리한다.

* 네임스페이스(namespace)

- 하나의 프로그램에서 변수나 함수를 여러 개 정의하다보면, 상황에 따라 이름이 중복될수 있다. 이런 중복을 피하는 방법이 네임스페이스다.

- 네임스페이스는 프로그램상에서 하나의 개체(변수, 함수)를 가리킬 수 있는 범위를 나타내는 공간이다. std 네임스페이스는 표준 라이브러리의 이름들이 정의 되어 있다.

- 동일한 함수 이름일지라도 서로 다른 네임스페이스에서는 다르게 정의될 수 있다.

|  |
| --- |
| using namespace std; |

* + - 의미: 표준 라이브러리 std에 정의된 이름의 변수나 함수를 사용하겠다.

- 네임스페이스의 일부 이름만 사용하고자 하는 경우

|  |
| --- |
| using std::cin;  using std::cout;  using std::endl; |

예제: ex01\_01.cpp에서 using namespace std; 수정

ex01\_03.cpp

|  |
| --- |
| #include <iostream>  using std::cin;  using std::cout;  using std::endl;  int main()  {  double radius, volume;  double PI = 3.14;  cout << "Enter a radius: ";  cin >> radius;  volume = (4.0 / 3) \* PI \* radius \* radius \* radius;  cout << "The volum is " << volume << endl;  return 0;  } |

1.4 변수, 식 및 할당문

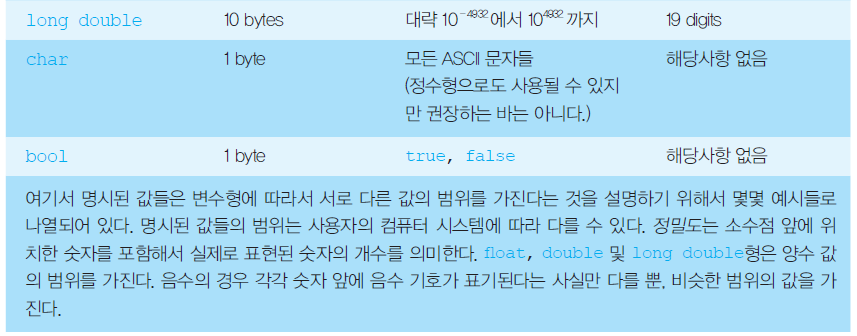
* 식별자(identifier)

- 하나의 함수 내에서 식별(구분)할 수 있는 ‘예약어+변수’를 의미한다.

- 변수는 문자 또는 언더바 기호로 시작한다.

* + - 언더바 식별자(변수)는 시스템 변수이다.
    - 관례상, 변수는 소문자로 시작하고, 단어와 단어 사이는 대문자를 사용한다. (topSeed)
    - 상수는 대문자로 선언한다.

bool 데이터형



- C++에만 있으며 논리식이나 조건문에서 사용한다

문자열 처리

- C에서는 문자형 배열을 선언하여 문자열을 처리했지만, C++에서는 string 클래스를 이용한다. string은 기본 자료형이 아니다. +연산자는 두 개의 문자열을 하나의 문자열로 만든다.

|  |
| --- |
| #include <string>  string question;  question = "Enter a radius: ";  cout << question; |

- cin >> string변수: 첫 번째 공백 문자를 만날 때까지 문자열을 string변수에 입력 받는다.

할당문 (변수 = 수식)

- l-value = r-value: l-value는 할당 연산자의 좌측 항에 표현된 값을 의미하고, r-value는 할당 연산자의 우측 항에 표현된 식을 통하여 정의된 값을 의미한다.

- 정수형 나눗셈

* + - 17/5 → 3
    - 17/5 → 3, (17%5) → 2

형 변환

- 자동 형 변환

* + - 17 / 5.5 → 17 정수는 자동적으로 17.0으로 형 변환이 이루어진다.

- 강제 형 변환 (C에서)

* + - 프로그래머가 형 변환 연산자를 사용하여 데이터 형을 변환시킨다.
    - (double)17 / 5.5 → 정수에 강제 형 변환 연산자를 사용
    - (double) myInt / myDouble → 변수에 강제 형 변환 연산자를 사용

- 강제 형 변환 (C++에서)

* + - static\_cast<자료형>(변수 또는 상수)

static\_cast<double>(17) / 5.5

상수에 const 선언

- 초기화된 변수의 값이 프로그램 상에서 변경되는 것을 방지하기 위해서 사용한다. 변경하려 하면 오류가 발생한다. const 등과 같은 예약어를 변경자 (modifier)라고 한다. 변수의 특성을 변경한다.

|  |
| --- |
| **const** double PI=3.14; |

예제: ex01\_01.cpp에서 문자열, static\_cast<자료형> 변환, const 선언

ex01\_04.cpp

|  |
| --- |
| #include <iostream>  #include<string>  using namespace std;  int main()  {  double radius, volume;  const double PI = 3.14;  string st1, st2, st3;  cin >> st1 >> st2 >> st3;  cout << st1 + " " + st2 + " " + st3 + ": ";  cin >> radius;  volume = (4.0 / static\_cast<double>(3)) \* PI \* radius \* radius \* radius;  cout << "The volume is " << volume << endl;  bool tf;  cout << "Is it true? ";  cin >> tf; // tf가 1이면 참, 그 외에는 거짓  if (tf == true) cout << "Nice! \n";  else cout << "Try again. \n";  return 0;  } |

증가 연산자

- 후위 증가 연산자

|  |
| --- |
| int n = 2, valueProduced;  valueProduced = 2\*(n++);  cout << valueProduced << endl;  cout << n << endl; |

결과: 4, 3

- 전위 증가 연산자

|  |
| --- |
| int n = 2, valueProduced;  valueProduced = 2\*(++n);  cout << valueProduced << endl;  cout << n << endl; |

결과: 6, 3

- 단독으로 사용되는 경우는 결과가 동일하다. n++; ++n;

퀴즈

- 다음 부분을 간소하게 하는 방법은?

|  |
| --- |
| volume = (4.0/3)\*PI\*radius\*radius\*radius; |

ex01\_05.cpp

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  int main()  {  double radius, volume;  double PI = 3.14;  cout << "Enter a radius: ";  cin >> radius;  volume = (4.0 / 3) \* PI \* pow(radius, 3);  cout << "The volum is " << volume << endl;  return 0;  } |

1.5 프로그램 스타일

- 내용 생략