3장 함수의 기본

- 함수는 프로시저(procedure), 서브프로그램, 메소드 라는 용어들로 사용되기도 한다.

- 함수의 형태

* 1개의 값을 리턴 하는 경우 예: int summation(int a, int b);
* 리턴 값이 없는 경우(void 함수) 예: void display(int a, int b);

3.1 사전 정의 함수

- C++에서 미리 만들어져 (사전에 정의되어) 있는 함수

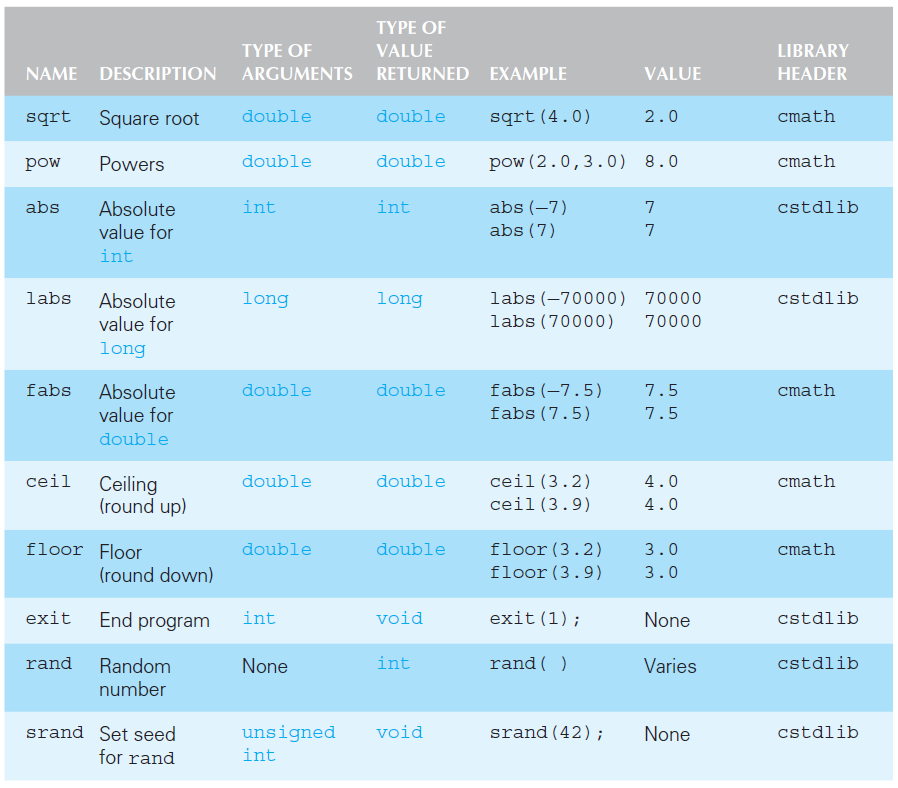
* 값을 리턴하는 사전 정의 함수

|  |
| --- |
| theRoot = sqrt(9.0);  thePower = pow(PI, 3.0); |

- 여기서 9.0는 함수의 인자(argument)이고 함수가 계산한 값(3.0)은 리턴 값이라 한다.

- 인자는 1개 이상 일 수 있지만, 리턴 값이 1개 보다 많은 함수는 없다.

- #include <cmath>에 있다.



* 사전에 정의된 void 함수

- void 함수는 어떤 행동을 수행하지만 값을 리턴하지 않는다.

- 식보다는 실행문으로 사용된다.

|  |
| --- |
| theResult = exit(1); // 오류  eixt(1); // 정상 |

- exit 함수를 호출하면 프로그램이 즉시 종료된다.

예제: 사전 정의 함수

ex03\_01.cpp

|  |
| --- |
| #include <iostream>  #include<cmath>  using namespace std;  int main()  {  /\*double f1, f2;  cout << "Enter two real numbers: ";  cin >> f1 >> f2;  cout << "square root: " << sqrt(f1) << endl;  cout << "powers : " << pow(f1, f2) << endl;  cout << "absolute : " << fabs(f1) << endl;  cout << "round up : " << ceil(f1) << endl;  cout << "round down : " << floor(f1) << endl;\*/  int n1;  long n2;  cout << "Enter two integer number: ";  cin >> n1 >> n2;    cout << "absolute : " << abs(n1) << endl;  exit(1);  cout << "absolute : " << labs(n2) << endl;    return 0;  } |

* 난수 생성기

- rand() 함수는 “무작위로 선택된 수”를 리턴하는 함수이다.

- #include <cstdlib>에 있다.

- 0 ~ RAND\_MAX(32767 이상)까지의 범위에 있는 정수를 리턴한다.

- 난수가 무작위로 선택되지만, 그 발생 순서가 동일한 의사 난수(pseudo random number) 이다.

* + - 프로그램 오류 수정 시, 오류가 있을 때와 같은 동일한 난수를 발생할 수 있어야만 오류 수정이 쉽다.

- 발생 순서를 결정하는 것을 시드(seed)라고 하고, 시드의 값을 조정하는 하는 함수는 srand(unsigned int)이다.

- 난수의 범위를 설정하기 위해서는 %, 나머지 연산자를 사용한다.

|  |
| --- |
| rand()%10; |

- 0.0 ~ 1.0 사이의 실수형 난수 생성

|  |
| --- |
| (RAND\_MAX-rand())/static\_cast<double>(RAND\_MAX); |

예제: 난수 생성기

ex03\_02.cpp

|  |
| --- |
| #include<iostream>  #include<cstdlib>  using namespace std;  int main()  {  srand(100); //실습2  for (int i = 0; i < 10; i++) //실습1  cout << rand() << endl;  for (int i = 0; i < 10; i++) //실습3  cout << (rand() % 10) << endl;    cout << "Maximum = " << RAND\_MAX << endl; //실습4  for (int i = 0; i < 10; i++)  cout << (RAND\_MAX - rand()) / static\_cast<double>(RAND\_MAX) << endl;  return 0;  } |

3.2 프로그래머 정의 함수

- 함수는 함수 선언(원형: prototype)과 함수 정의로 나눌 수 있다.

- 함수는 호출되기 전에 함수 선언이나 정의가 나타나야 한다.

- main 함수 앞에 함수 정의가 있으면 함수의 선언은 생략해도 된다.

함수 선언

- 함수의 이름과 사용된 인자의 수와 종류를 말해준다.



// 함수 선언에서만 사용

함수 정의

- 함수 헤더와 함수 본체로 구성된다.



- 함수의 본체 안에 다른 함수를 정의해서는 안 된다.

예제: 반올림 함수

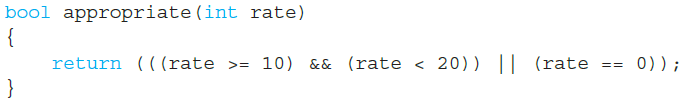
ex03\_03.cpp

|  |
| --- |
| #include<iostream>  #include<cmath>  using namespace std;  //int roundoff(double dv); //int roundoff(double); //실습3  void roundoff(double dv);  int main()  {  //--- 정수형 반올림 계산: 실습1  /\*double dv;  cout << "Enter a number: ";  cin >> dv;  cout << "Round off number: ";  cout << static\_cast<int>(floor(dv + 0.5)) << endl;\*/  //--- 정수형 반올림 반복 계산: 실습2  /\*double dv;  char ans;  do {  cout << "Enter a number: ";  cin >> dv;  cout << "Round off number: ";  cout << static\_cast<int>(floor(dv + 0.5)) << endl;  cout << "Again? (y/n): ";  cin >> ans;  } while (ans == 'y' || ans == 'Y');\*/  //--- 정수형 반올림 반복 함수: 실습3  /\*double dv;  char ans;  do {  cout << "Enter a number: ";  cin >> dv;  cout << "Round off number: ";  cout << roundoff(dv) << endl;  cout << "Again? (y/n): ";  cin >> ans;  } while (ans == 'y' || ans == 'Y');\*/  //--- void형 반올림 반복 함수: 실습4  double dv;  char ans;  do {  cout << "Enter a number: ";  cin >> dv;  cout << "Round off number: ";  roundoff(dv);  //cout << roundoff(dv) << endl;  cout << "Again? (y/n): ";  cin >> ans;  } while (ans == 'y' || ans == 'Y');  return 0;  }  //int roundoff(double dv) //정수형 반올림 반복 함수: 실습3  //{  // /\*int res;  // res = static\_cast<int>(floor(dv + 0.5));  // return res;\*/  // return static\_cast<int>(floor(dv + 0.5));  //}  void roundoff(double dv) //void형 반올림 반복 함수: 실습4  {  /\*int res;  res = static\_cast<int>(floor(dv + 0.5));  return res;\*/  cout << static\_cast<int>(floor(dv + 0.5)) << endl;  } |

* 여러 가지 함수

부울 값을 리턴하는 함수

- 함수의 리턴형으로 bool이 될 수 있다.



void 함수

- 리턴형 자리에 void 키워드를 사용한다.

- return 문이 필요 없다.

main 함수

- 시스템만이 호출할 수 있다.

- main 함수를 재귀적으로 호출할 수 없다.

3.3 영역 규칙

* 함수의 추상화

- 함수(작은 프로그램)가 무엇을 하는가(what)는 알아야 하지만, 어떻게 실행되는 가(how)는 알 필요가 없도록 프로그램을 작성하는 것을 추상화라고 한다.

- 다른 표현으로는 정보 은닉(information hiding), 블랙박스화(black box) 라고도 한다.

- 추상화 작성 방법

* + - 프로그래머가 알아야 할 것은 함수 선언과 이 선언에 표시된 주석이 전부여야 한다.
    - 함수 선언 주석에 함수 인자에 대한 모든 요구 조건을 말하고, 함수 호출 결과에 대해 기술해야 한다.
    - 함수 본체에서 사용하는 모든 변수는 지역 변수이어야 한다. (선택적 추천)
* 변수와 영역

지역(local) 변수

- 함수의 정의 내부에 선언된 변수를 지역 변수라고 하고, 해당 함수가 변수의 영역(scope)이 된다.

전역(global) 변수

- 프로그램의 시작 부분, 모든 함수의 본체 밖에 선언된 변수를 의미한다.

- 프로그램의 추상화와 재사용을 위해 되도록이면 사용하지 않는다.

블록

- 중괄호{}로 묶은 복합문을 블록이라 한다.

- 블록 안에 선언된 변수는 이 블록에 지역적이다.

예제: 반지름을 입력하여 원의 넓이와 구의 부피를 구하라.

ex03\_04.cpp

|  |
| --- |
| #include<iostream>  #include<cmath>  using namespace std;  //double area(double PI, double rd); // 실습 2  //double Volume(double PI, double rd);  void area(double PI, double rd); // 실습 3  void Volume(double PI, double rd);  int main()  {  //--- 함수를 사용하지 않는 경우: 실습1  // double rd, ar, vl;  // const double PI = 3.14159;  //  // cout << " Enter a radius: ";  // cin >> rd;  //  // ar = PI \* pow(rd, 2);  // vl = (4.0 / 3.0) \* PI \* pow(rd, 3);  //  // cout << "Radius = " << rd << endl  // << "Area = " << ar << endl  // << "Volume = " << vl << endl;  //--- 반환값이 있는 함수: 실습2  // double rd, ar, vl;  // const double PI = 3.14159;  //  // cout << "Enter a radius: ";  // cin >> rd;  //  // ar = area(PI, rd); // PI\* pow(rd, 2);  // vl = Volume(PI, rd); // (4.0 / 3.0)\* PI\* pow(rd, 3);  //  // cout << "Radius = " << rd << endl  // << "Area = " << ar << endl  // << "Volume = " << vl << endl;  //--- void 함수: 실습3  double rd, ar, vl;  const double PI = 3.14159;  cout << "Enter a radius: ";  cin >> rd;  cout << "Radius = " << rd << endl;  area(PI, rd);  Volume(PI, rd);  return 0;  }  //--- 반환값이 있는 함수: 실습2  //double area(double a, double b) // 실습 2 // area(double PI, rd)  //{  // return a \* pow(b, 2);  //}  //  //double Volume(double PI, double rd)  //{  // return (4.0 / 3.0) \* PI \* pow(rd, 3);  //}  //--- void 함수: 실습3  void area(double a, double b) // 실습 3  {  double ar;  ar = a \* pow(b, 2);  cout << "Area = " << ar << endl;  }  void Volume(double PI, double rd)  {  cout << "Volume = " << (4.0 / 3.0) \* PI \* pow(rd, 3) << endl;  } |