제1장 기초 사항

이번 장에서 학습할 내용

- C++ 개요
- 객체 지향의 간단한 소개
- 통합개발환경
- 상수, 변수, 자료형
- 변수와 상수
- 연산자

C++에 대한 기초적인 사항들을 살펴봅니다.

C++ 개요

- C++는 1980년대 초에 AT&T 벨연구소의 Bjarne Stroustrup이 개발
- C++는 C언어를 유지, 확장한 것
- C with Classes -> C++
- C++는 C언어에 클래스 개념, 가상 함수, 연산자 중복 정의, 다중 상속, 템플릿, 예외 처리 등이 기능이 차례로 추가

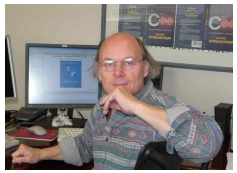


그림 1.4 C++는 3가지의 프로그래밍 방법을 지원한다.

C++의 설계 철학

- 엄격한 타입 검사, 범용 언어, 효율적, 이식성
- 여러 가지의 프로그래밍 스타일을 지원 (절차 지향 프로그래밍, 데이터 추상화, 객체 지향 프로그래밍, 일반화 프로그래밍)
- 프로그래머가 자유롭게 선택할 수 있도록 설계
- 최대한 C와 호환
- 플랫폼에 의존적이거나 일반적이지 않은 특징은 제거

절차적/구조화/객체지향 프로그래밍

- 절차적 프로그래밍
  - 데이터보다는 알고리즘(절차)을 중시
  - GOTO문이나 JUMP
- 구조적 프로그래밍(structured programming)
  - 절차적 프로그래밍의 하위 개념
  - GOTO문을 없애거나 GOTO문에 대한 의존성을 줄여줌
  - 여전히 데이터와 알고리즘은 분리되어 있음
- 객체 지향 프로그래밍(object-oriented programming)
  - 데이터와 알고리즘이 묶여있음

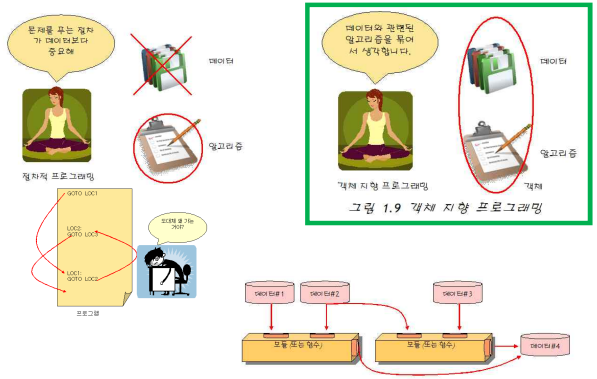


그림 1.9 객체 지향 프로그래밍

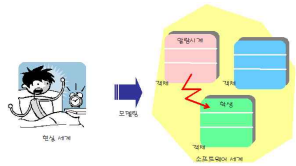
그림 1.8 구조화 프로그래밍에서도 데이터와 알고리즘은 분리되어 있다.

절차적/구조화/객체지향 프로그래밍

- 프로그래밍 사례 : 구구단 프로그램
  - 절차적 프로그래밍
  - 구조적 프로그래밍
  - 객체 지향 프로그래밍

객체 지향 프로그래밍이란?

- 실제 세계를 모델링하여 소프트웨어를 개발하는 방법
- 주요 개념
  - 캡슐화(encapsulation)
  - 정보 은닉(information-hiding)
  - 상속(inheritance)
  - 다형성(polymorphism)



  - 데이터와 알고리즘을 하나의 단위(클래스)로 묶는 것이다.
  - 데이터에 대한 불필요한 접근을 차단하여서 데이터를 보호
  - 비슷한 클래스가 이미 존재하고 있다면 그 클래스를 가져다가 사용하는 것
  - 같은 이름의 함수나 연산자를 중복 정의하여서 상황에 따라서 가장 적절한 함수나 연산자를 프로그램이 자동적으로 선택

- 객체 지향의 최종 목표: 코드의 재사용

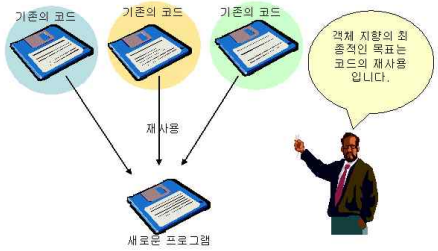


그림 1.11 객체 지향 프로그래밍의 최종목표는 코드의 재사용

- 객체 지향의 최종 목표: 코드의 재사용

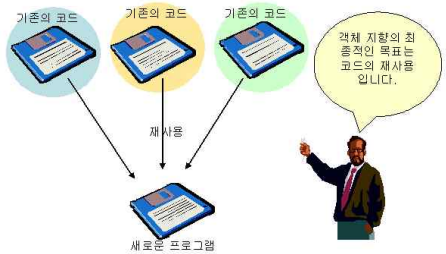


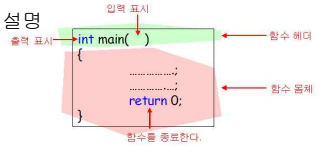
그림 1.11 객체 지향 프로그래밍의 최종목표는 코드의 재사용

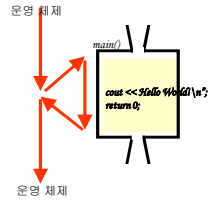
통합 개발 환경

- 통합 개발 환경(IDE: integrated development environment):
  - 에디터 + 컴파일러 + 디버거
- 용어
  - 솔루션
  - 프로젝트
  - 소스 파일(헤더파일, cpp파일)
  - 인라인 함수
  - 컴파일, 링크, 디버깅
  - 프로그램 실행

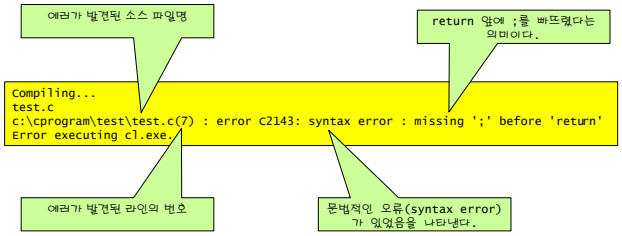
Game Over Game

- 주석(comment): 프로그램에 대한 설명
- 헤더 파일 포함
  - #include <iostream>
  - using namespace std;
- 함수
  - int main()
- 문장
  - 함수는 여러 개의 문장으로 이루어진다.
  - 문장들은 순차적으로 실행된다.
- 출력 객체 cout
  - cout << “Hello World!” << endl;



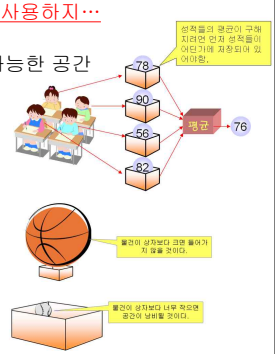


오류 메시지의 분석



상수, 변수, 자료형

- 변수(variable): 저장된 값의 변경이 가능한 공간
  - 프로그램에서 일시적으로 데이터를 저장하는 공간
  - 데이터를 어딘가에 저장해야 다음에 사용하지…
- 상수(constant): 저장된 값의 변경이 불가능한 공간
  (예) 3.14, 100, 'A', "Hello World!"
- l-value, r-value
- **자료형(data type)**: 데이터의 타입(종류)
  - (예) 정수형 , 실수형,…



자료형의 종류

- 자료형은 너무너무너무너무너무너무 중요함!!!

자료형			설명	바이트수	범위
정수형	부호있음	short	short형 정수	2	-32768 ~ 32767
		int	정수	4	-2147483648 ~ 2147483647
		long	long형 정수	4	-2147483648 ~ 2147483647
	부호없음	unsigned short	부호없는 short형 정수	2	0 ~ 65535
		unsigned int	부호없는 정수	4	0 ~ 4294967295
		unsigned long	부호없는 long형 정수	4	0 ~ 4294967295
문자형	부호있음	char	문자 및 정수	1	-128 ~ 127
	부호없음	unsigned char	문자 및 부호없는 정수	1	0 ~ 255
부동소수점형		float	단일정밀도 부동소수점	4	1.2E-38 ~ 3.4E38
		double	두배정밀도 부동소수점	8	2.2E-308 ~ 1.8E308
부울형		bool	참이나 거짓을 나타낸다.	1	true, false

이름짓기

- 이름을 잘 짓자!
  - 변수, 상수, 함수 이름
  - 클래스 이름
  - 솔루션 이름, 프로젝트 이름, 소스파일 이름
- 규칙
  - 알파벳 문자와 숫자, 밑줄 문자 _로 구성
  - 첫 번째 문자는 반드시 알파벳 또는 밑줄 문자 _
  - 대문자와 소문자를 구별
  - 키워드와 똑같은 이름은 허용되지 않는다.

키워드

- 키워드(keyword): C++언어에서 고유한 의미를 가지는 특별한 단어
- 예약어(reserved words) 라고도 한다.

C언어의 키워드

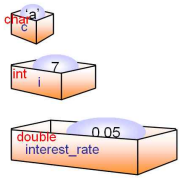
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

C++언어의 키워드

asm	false	protected	try
bool	friend	public	typeid
catch	inline	reinterpret_cast	typename
class	mutable	static_cast	using
const_cast	namespace	template	virtual
delete	new	this	wchar_t
dynamic_cast	operator	throw	
explicit	private	true	

변수 선언

- 변수 선언: 컴파일러에게 어떤 변수를 사용하겠다고 미리 알리는 것
- 변수 선언, 초기화의 예
  - char c;
  - int i;
  - double interest_rate;
  - int height, width;
  - char c = 'a';
  - int i = 7;
  - double interest_rate = 0.05;



- 기호 상수(symbolic constant): 기호를 이용하여 상수를 표현한 것
  - area = 3.141592 * radius * radius;
  - area = PI * radius * radius;
  - #define PI 3.141592
  - const int PI = 3.141592;

문자열 클래스

- string 타입을 제공한다.

string.cpp

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

int main(void)
{
   string s1 = "Good";
   string s2 = "Morning";
   string s3 = s1 + " " + s2 + "!\n";
   cout << s3;
   return 0;
}
```

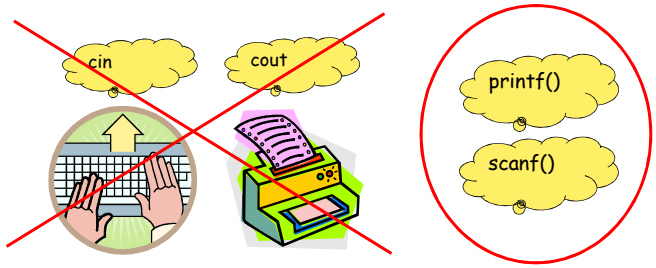
string 헤더파일을 반드시 포함하여야 한다.

실행결과

```
Good Morning!
```


출력과 입력

- C++에서는 콘솔 입력은 cin 객체가, 콘솔 출력은 cout 객체가 담당
- 이들은 모두 iostream 라이브러리에 포함

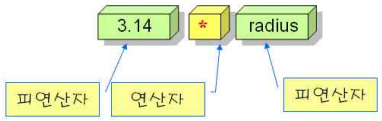


수식과 연산자

- 수식(expression)

```
x + y
x*x + 5*x + 6
(principal * interest_rate * period) / 12.0
```

- 수식(expression)
  - 상수, 변수, 연산자의 조합
  - 연산자와 피연산자로 나누어진다.



연산자의 분류

연산자의 분류	연산자	의미
대입	=	오른쪽을 왼쪽에 대입
산술	+ - * / %	사칙연산과 나머지 연산
부호	+ -	
증감	++ --	증가, 감소 연산
관계	> < == != >= <=	오른쪽과 왼쪽을 비교
논리	&& || !	논리적인 AND, OR
조건	?	조건에 따라 선택
콤마	,	피연산자들을 순차적으로 실행
비트 단위 연산자	& | ^ ~ << >>	비트별 AND, OR, XOR, 이동, 반전
sizeof 연산자	sizeof	자료형이나 변수의 크기를 바이트 단위로 반환
형변환	(type)	변수나 상수의 자료형을 변환
포인터 연산자	* & []	주소계산, 포인터가 가리키는 곳의 내용 추출
구조체 연산자	. ->	구조체의 멤버 참조

단항, 이항, 삼항

- 단항 연산자: 피연산자의 수가 1개

```
++x;
--y;
```

- 이항 연산자: 피연산자의 수가 2개

```
x + y
x - y
```

- 삼항 연산자: 연산자의 수가 3개

```
x ? y : z
```


산술, 관계, 논리

- 산술 연산자

연산자	기호	의미	예
덧셈	+	x와 y를 더한다	x+y
뺄셈	-	x에서 y를 뺀다.	x-y
곱셈	*	x와 y를 곱한다.	x*y
나눗셈	/	x를 y로 나눈다.	x/y
나머지	%	x를 y로 나눌 때의 나머지값	x%y

- 관계 연산자

연산자 기호	의미	사용예
==	x와 y가 같은가?	x == y
!=	x와 y가 다른가?	x != y
>	x가 y보다 큰가?	x > y
<	x가 y보다 작은가?	x < y
>=	x가 y보다 크거나 같은가?	x >= y
<=	x가 y보다 작거나 같은가?	x <= y

- 논리 연산자

연산자 기호	사용예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
||	x || y	OR 연산, x나 y중에서 하나만 참이면 참, 아니면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

논리 연산자

- 여러 개의 조건을 조합하여 참과 거짓을 따지는 연산자
- 결과값은 참(1) 아니면 거짓(0)

연산자 기호	사용예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
||	x || y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

연산자 우선 순위 + 결합방향

- 수식에서 어떤 연산자를 먼저 계산할 것인지의 문제

우선 순위	연산자	결합 규칙
1	() [] -> . ++(후위) --(후위)	->(좌에서 우)
2	sizeof &(주소) ++(전위) --(전위) ~ ! *(역참조) +(부호) -(부호), 형변환	<-(우에서 좌)
3	*(곱셈) / %	->(좌에서 우)
4	+(덧셈) -(뺄셈)	->(좌에서 우)
5	<< >>	->(좌에서 우)
6	< <= >= >	->(좌에서 우)
7	== !=	->(좌에서 우)
8	&(비트연산)	->(좌에서 우)
9	^	->(좌에서 우)
10	|	->(좌에서 우)
11	&&	->(좌에서 우)
12	||	->(좌에서 우)
13	?(삼항)	->(우에서 좌)
14	= += *= /= %= &= ^= |= <<= >>=	->(우에서 좌)
15	,(콤마)	->(좌에서 우)

우선 순위의 일반적인 지침

- 콤마 < 대입 < 논리 < 관계 < 산술 < 단항
- 괄호 연산자는 가장 우선순위가 높다.
- 모든 단항 연산자들은 이항 연산자들보다 우선순위가 높다.
- 콤마 연산자를 제외하고는 대입 연산자가 가장 우선순위가 낮다.
- 관계 연산자나 논리 연산자는 산술 연산자보다 우선순위가 낮다.
  - x + 2 == y + 3

- 연산자들의 우선 순위가 생각나지 않으면 괄호를 이용
  - (x <= 10) && (y >= 20)
  - x = y = z = k = 2;

시험에 많이 낼 문제

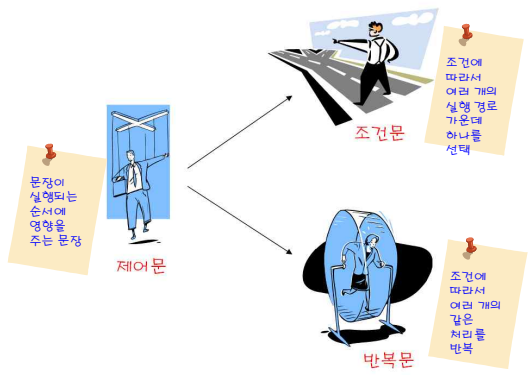
- 연산의 순서와 자료형

```
result = x * y % z;
         ① ②
```

```
int x=0, y=0, z=0;

cout << (2 > 3 || 6 > 7) << endl;
cout << (2 || 3 && 3 > 2) << endl;
cout << (x = y = z = 1) << endl;
cout << (- ++x + y--) << endl;
```

```
0
1
1
-1
```

제어문



if-else 문

```
if ( score >= 60 )
    cout << "합격입니다.\n";
else
    cout << "불합격입니다.\n";
```

score가 60이상이면 실행

score가 60미만이면 실행

```
if ( score >= 60 )
{
    cout << "합격입니다.\n";
    cout << "장학금도 받을 수 있습니다.\n";
}
else
{
    cout << "불합격입니다.\n";
    cout << "공부하세요.\n";
}
```

score가 60이상이면 실행

score가 60미만이면 실행

if와 else의 매칭 문제

else 절은 가장 가까운 if절과 매치된다.

```
if(score >= 80)
    if( score >= 90)
        cout << "당신의 학점은 A입니다\n";
    else
        cout << "당신의 학점은 B입니다\n";
```

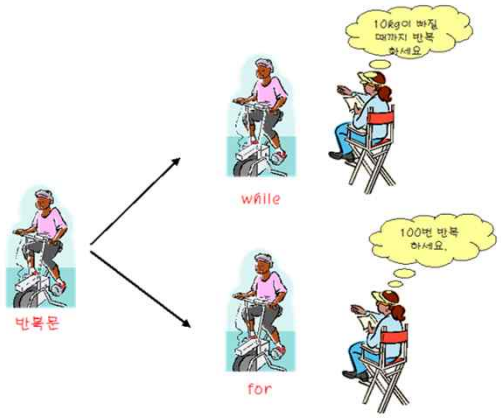

switch - case문

```
int main()
{
    int number;

    cout << "정수를 입력하시오:";
    cin >> number;
    switch(number)
    {
        case 0:
            cout << "없음\n";
            break ;
        case 1:
            cout << "하나\n";
            break ;
        case 2:
            cout << "둘\n";
            break ;
        default:
            cout << "많음\n";
            break;
    }
}
```

number → 0, 1, 2, 기타

정수를 입력하시오: 1
하나

반복문의 종류



while 문

```
// while 문을 이용한 구구단 출력 프로그램
#include <iostream>
using namespace std;

int main()
{
    int n;
    int i = 1;

    cout << "구구단 중에서 출력하고 싶은 단을 입력하시오: ";
    cin >> n;
    while (i <= 9){
        cout << n << "*" << i << "=" << n * i << endl;
        i++;
    }
    return 0;
}
```

```
구구단 중에서 출력하고 싶은 단을 입력하시오: 9
9*1 = 9
9*2 = 18
9*3 = 27
....
9*9 = 81
```

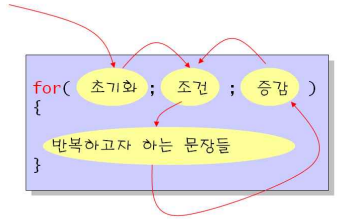

do...while문

```
int main()
{
        int i = 10;
        do {
            cout <<"i의 값: " << i << endl;
            i++;
        } while (i <3);
}
```

```
i의 값: 10
```


for 문

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

int main()
{
    long fact = 1;
    int n;

    cout << "정수를 입력하시오:";
    cin >> n;

    for (int i = 1; i <= n; i++)
        fact = fact * i;

    cout << n << "!은 " << fact << "입니다.\n";

    return 0;
}
```




```
정수를 입력하시오: 10
10!은 3628800입니다.
```

게임 예제

```
// 반복을 이용한 게임 프로그램
#include <iostream>
using namespace std;

int main()
{
        int answer =59;        // 정답
        int guess;
        int tries = 0;
        // 반복구조
        do {
                cout << "정답을 추측하여 보시오: ";
                cin >> guess;
                tries++;

                if (guess >answer)      // 사용자가 입력한 정수가 정답보다 높으면
                        cout <<"제시한 정수가 높습니다.\n";
                if (guess <answer)      // 사용자가 입력한 정수가 정답보다 낮으면
                        cout <<"제시한 정수가 낮습니다.\n";
        } while (guess !=answer);

        cout << "축하합니다. 시도횟수=" << tries << endl;

        return 0;
}
```

```
정답을 추측하여 보시오: 10
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 30
제시한 정수가 낮습니다.
정답을 추측하여 보시오: 60
제시한 정수가 높습니다.
정답을 추측하여 보시오: 59
축하합니다. 시도횟수=4
```


break 문과 continue 문

- break 문은 반복 루프를 빠져 나오는데 사용된다.
- continue 문은 현재 수행하고 있는 반복 과정의 나머지를 건너뛰고 다음 반복 과정을 강제적으로 시작
- 다중 루프를 빠져나오려면?

프로그래밍 프로젝트

- 시큰둥한 게임 만들기
  - Game Over 게임 v2 (Ascii Art)
  - 구구단 게임
  - 섭씨 화씨 변환 게임
  - 이차방정식 근의 공식
  - 소수 찾기
  - 세금 계산

Q & A