객체지향프로그래밍 (202004100 장주훈, 202084023 이병길)

**13주차 – C++ 입출력 시스템**

* 스트림(stream): 데이터의 흐름, 혹은 데이타를 전송하는 소프트웨어 모듈
* C++ 스트림 종류
* 입력 스트림
* 출력 스트림

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

키 입력 스트림의 버퍼: 입력장치로부터 입력된 데이터를 프로그램으로 전달하기 전에 일시 저장

+ <Enter> 키가 입력된 시점부터 키 입력 버퍼에서 프로그램이 읽기 시작

* + 스트림 입출력 방식(stream I/O)
    - * 버퍼가 꽉 차거나, ‘\n’을 만나거나, 강제 출력 명령의 경우에만 버퍼가 출력 장치에 출력
  + 저 수준 입출력 방식(raw level console I/O)
    - 키가 입력되는 즉시 프로그램에게 키 값 전달
* <Backspace>키 그 자체도 프로그램에게 바로 전달
* 게임 등 키 입력이 즉각적으로 필요한 곳에 사용
  + - 프로그램이 출력하는 즉시 출력 장치에 출력
    - 컴파일러마다 다른 라이브러리나 API 지원

다양한 크기의 다국어 문자를 수용하기 위해, 입출력 라이브러리가 템플릿으로 작성됨

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

C++ 표준 입출력 스트림 객체

* + Cerr: 오류 메시지를 출력할 목적(스트림 내부 버퍼 거치지 않고 출력)
  + Clog: 오류 메시지를 출력할 목적(스트림 내부에 버퍼 거쳐 출력)

텍스트, 폰트, 라인, 화이트보드이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| #include <iostream>  using namespace std;  void get1() {  cout << "cin.get()로 <Enter> 키까지 입력 받고 출력합니다>>";  int ch; // EOF와의 비교를 위해 int 타입으로 선언  while ((ch = cin.get()) != EOF) { // 문자 읽기. EOF 는 -1  cout.put(ch); // 읽은 문자 출력  if (ch == '\n')  break; // <Enter> 키가 입력되면 읽기 중단  }  }  void get2() {  cout << "cin.get(char&)로 <Enter> 키까지 입력 받고 출력합니다>>";  char ch;  while (true) {  cin.get(ch); // 문자 읽기  if (cin.eof()) break; // EOF를 만나면 읽기 종료  cout.put(ch); // ch의 문자 출력  if (ch == '\n')  break; // <Enter> 키가 입력되면 읽기 중단  }  }  int main() {  get1(); // cin.get()을 이용하는 사례  get2(); // cin.get(char&)을 이용하는 사례  } |

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

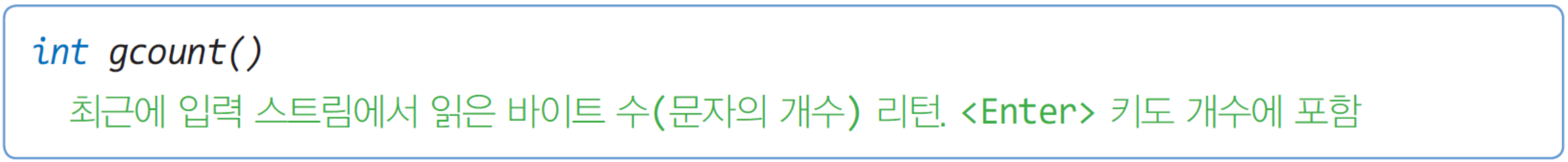
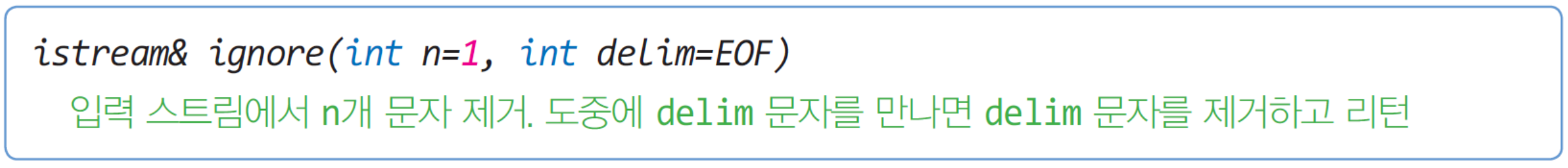
자동 생성된 설명

**!주의**

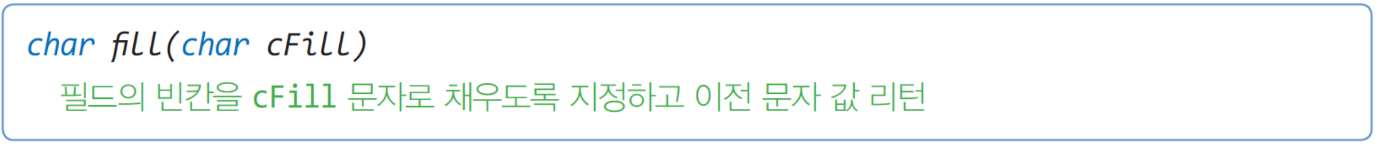
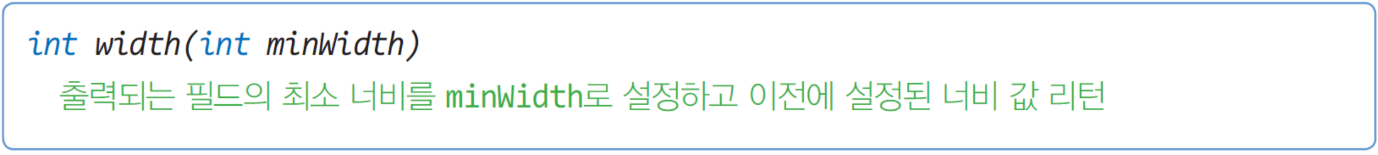
* + **<Enter> 키(‘\n’)는 스트림 버퍼에 남아 있음**
    - 다시 get()으로 문자열 읽기를 시도하면 입력 스트림에 남은 ‘\n’키를 만나게 되어바로 리턴. 프로그램은 무한 루프에 빠질 수 있음
    - 이때 이 문제를 해결하기 위해서는 cin.get()이나 cin.ignore(1);를 통해 문자 1개(‘\n’)를 스트림에서 읽어 버려야 함.

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

조작자

* + - C++ 표준 라이브러리에 구현된 조작자 : 입출력 포맷 지정 목적
    - 개발자 만의 조작자 작성 가능 : 다양한 목적
    - 매개 변수 없는 조작자와 매개 변수를 가진 조작자로 구분

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* 삽입 연산자(<<)

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

사용자 삽입 연산자 만들기

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

Cout<<p == cout .<< (point p)

|  |
| --- |
| #include <iostream>  using namespace std;  class Point { // 한 점을 표현하는 클래스  int x, y; // private 멤버  public:  Point(int x = 0, int y = 0) {  this->x = x;  this->y = y;  }  friend ostream& operator << (ostream& stream, Point a);  };  // << 연산자 함수  ostream& operator << (ostream& stream, Point a) {  stream << "(" << a.x << "," << a.y << ")";  return stream;  }  int main() {  Point p(3, 4); // Point 객체 생성  cout << p << endl; // Point 객체 화면 출력  Point q(1, 100), r(2, 200); // Point 객체 생성  cout << q << r < endl; // Point 객체들 연속하여 화면 출력  } |

프렌드 함수 사용 이유: 클래스 멤버 함수로는 적합하지 않지만, 클래스의 protected, private 멤버를 접근해야 하는 특별한 경우 사용(연산자 함수에서 주로 사용)

연산자 중복

텍스트, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명

Cout.<<(point p) – c++에서 제공 안함.

<<cout (stream ,point p) – 함수 연산자에 함수를 작성해주면 사용 가능

* 추출 연산자(>>)

|  |
| --- |
| #include <iostream>  using namespace std;  class Point { // 한 점을 표현하는 클래스  int x, y; // private 멤버  public:  Point(int x = 0, int y = 0) {  this->x = x;  this->y = y;  }  friend istream& operator >> (istream& ins, Point& a); // friend 선언  friend ostream& operator << (ostream& stream, Point a); // friend 선언  };  istream& operator >> (istream& ins, Point& a) { // >> 연산자 함수  cout << "x 좌표>>";  ins >> a.x;  cout << "y 좌표>>";  ins >> a.y;  return ins;  }  ostream& operator << (ostream& stream, Point a) { // << 연산자 함수  stream << "(" << a.x << "," << a.y << ")";  return stream;  }  int main() {  Point p; // Point 객체 생성  cin >> p; // >> 연산자 호출하여 x 좌표와 y 좌표를 키보드로 읽어 객체 p 완성  cout << p; // << 연산자 호출하여 객체 p 출력  } |

사용자 조작자 만들기

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

Endl은 포인터 함수로 c++에서 만들어 둔 조작자이므로 같은 원리로 사용자 지정 조작자를 생성 가능하다.

Flush는 버퍼를 비우도록 하는 것.

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명