프로그래밍언어론

문 1. 예외 처리(Exception Handling) 기능이 있는 프로그래밍 언어로 프로그램을 작성하면 어떤 장점이 있는가?

① 신뢰도(reliability)가 높은 프로그램을 작성할 수 있다.
② 컴파일러에게 예외가 일어나는 부분을 알려줌으로써 빠른 수행이 가능한 기계어를 생성할 수 있다.
③ 예외 처리를 주프로그램과 병행처리할 수 있기 때문에 빠른 수행이 가능하다.
④ 예외 처리는 동적으로 발생하는 사건이기 때문에 인터프리터 형식으로 구현되는 언어에서만 그 기능을 제공할 수 있다.

문 2. 프로그래밍 언어에서 자료형(data type)을 사용하는 목적으로 가장 거리가 먼 것은?

① 번역기가 자료형 정보를 이용하여 기억 장소를 효율적으로 할당할 수 있다.
② 데이터와 이에 적용되는 연산의 적합성을 분석함으로써, 프로그램에 내재한 오류를 번역시에 미리 검출할 수 있다.
③ 프로그램을 동적으로 재구성하여 그 결과값을 미리 알 수 있게 됨으로써, 신속한 프로그램 완성이 가능하다.
④ 사용자 정의 자료형(user-defined data type)을 사용하게 되면, 프로그램의 신뢰성(reliability)과 가독성(readability)을 향상시킬 수 있다.

문 3. 다음은 C 언어로 작성된 간단한 프로그램이다.

```
main() {
      int x;
      double y;
      x = 1 + 2;
      y = x + 4.0;
}
```

주어진 프로그램에 대한 다음 설명 중에서 적절한 것만을 모두 고른 것은?

ㄱ. 암묵적인(implicit) 자동 자료형 변환(automatic type conversion)이 수행된다.
ㄴ. 타입 오류(type error)가 발생한다.
ㄷ. 중복 정의(overload)된 연산자(operator)가 사용되었다.
ㄹ. 정적 자료형 바인딩(static type binding)이 사용되었다.

① ㄱ, ㄹ
② ㄴ, ㄷ
③ ㄱ, ㄴ, ㄷ
④ ㄱ, ㄷ, ㄹ

문 4. 다음 C++ 프로그램의 실행 결과는?

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

int main( )
{    int *ip, d[5] = {1, 2, 3, 4, 0};
     int index;
     ip = d+1;
     for(index = 0; index < 3; index++)
     {
          d[index+2] = *(ip+1) + *ip - 1;
          ip = ip+1;
     }
     for(int i = 0; i < 5; i++) cout << d[i] << " ";
     return 0;
}
```

① 1 2 3 5 7
② 1 2 4 7 11
③ 1 2 4 7 6
④ 1 2 3 6 7

문 5. C와 유사한 언어로 작성된 다음 프로그램의 출력 결과는? [단, 정적 영역규칙(static scope rule)과 참조 전달 방식(pass by reference)의 함수 호출을 사용한다고 가정한다]

```
int x = 1, y = 2;
void a(int z) {
  z = x+2*y-z;
}
void main() {
  int x = 3, y = 4;
  a(y); a(x);
  printf("%d", x);
}
```

① 1
② 2
③ 3
④ 14

문 6. C 언어는 논리곱(logical AND ; &&)과 논리합(logical OR ; ||) 연산자(operator)에 대해 '단락-회로 평가(short-circuit evaluation)'를 하도록 되어 있다. 아래 C 프로그램의 수행 결과는?

```
int main()
{
   int a = 1, b = 2, c = 3, d = 4;
   if ((a == b)&&(c++ == d)) c++;
   printf("%d", c);
}
```

① 3
② 4
③ 5
④ 6

문 7. 스칼라(scalar) 변수는 기억장소 바인딩과 존속기간에 따라 정적(static), 스택-동적(stack-dynamic), 명시적 힙-동적(explicit heap-dynamic), 묵시적 힙-동적(implicit heap-dynamic)으로 구분할 수 있다. 이들에 대한 설명으로 옳지 않은 것은?

① 정적 변수는 실행 전에 기억장소에 바인딩이 이루어지고 이것이 프로그램 실행 끝까지 유지되는 변수이다.

② 스택-동적 변수는 그 변수가 선언된 함수가 호출될 때 바인딩이 이루어지며, 과거 민감(history sensitive) 변수이다. 예로는 C++ 함수의 내부에서 선언된 변수가 있다.

③ 명시적 힙-동적 변수는 프로그래머에 의해서 명세되는 명시적 실행 시간 명령어에 의해서 할당되는 이름 없는 메모리 셀이다. 주로 포인터를 사용하여 참조되며 C++에서는 new 연산자를 사용하여 할당될 수 있다.

④ 묵시적 힙-동적 변수는 배정문(assignment statement) 형식으로 값이 변수에 배정될 때 힙 기억장소에 바인딩되며 JavaScript와 같은 언어에서 사용된다.

문 8. 다음 C++ 코드의 출력 결과는?

```
class X{
public: virtual void a() {cout<< "a of X"<<endl;}
        void b() {cout<< "b of X"<<endl;}
        virtual void c() {a(); b();}
};

class Y : public X{
public: void a() {cout<< "a of Y"<<endl;}
};

class Z : public Y{
public: void b() {cout<< "b of Z"<<endl;}
        void c() {b(); a();}
};
...
X x1; Z z1;
x1 = z1; x1.c();
X* ap = new Y; ap->c();
ap = new Z; ap->c();
```

① a of X
b of X
a of Y
b of X
b of Z
a of Y

② a of X
b of Z
a of X
b of X
b of Z
a of Y

③ b of Z
a of X
a of Y
b of X
b of Z
a of Y

④ b of X
a of X
a of X
b of X
b of Z
a of Y

문 9. 다음은 Java로 작성된 프로그램이다. 이에 대한 설명으로 가장 부적절한 것은?

```
class People {
    public void hello() {
        System.out.println("Hello !"); }
    public void introduce() {
        System.out.println("Hi !"); }
}
class Student extends People {
    public void introduce() { hello(); }
}
class Main {
   public static void main(String args) {
      People p = new People();
      Student s = new Student();
      call(p);
      call(s);
   }
   static void call(People p) { p.introduce(); }
}
```

〈실행결과〉
Hi !
Hello !

① call 메소드의 정의와 사용에서 다형성(polymorphism) 기법이 사용되었다.

② call 메소드의 정의와 사용에서 동적 바인딩(dynamic binding) 기법이 사용되었다.

③ Student 클래스의 정의에서 상속(inheritance) 기법이 사용되었다.

④ Student 클래스의 정의에서 중복정의(overloading) 기법이 사용되었다.

문 10. 객체지향 언어에서 어떤 객체에 메시지를 보내면 그 메시지에 해당되는 메소드를 찾아서 수행하여야 한다. 이와 같은 메시지와 메소드 사이의 바인딩에 대한 설명 중 옳은 것은?

① Smalltalk와 같은 언어에서는 수행 속도를 빠르게 하기 위하여 컴파일시에 이런 바인딩을 모두 수행한다.

② C++에서 가상함수(virtual function) 이외의 모든 멤버 함수는 동적으로 바인딩 된다.

③ 객체지향 언어의 특징 중의 하나인 다형성(polymorphism)을 제공하기 위해서는 이런 바인딩이 수행시에 일어나야 한다.

④ C++ 언어에서는 단일 상속(single inheritance)인 경우 컴파일시에 바인딩을 하고, 다중 상속(multiple inheritance)인 경우 수행시에 바인딩을 한다.

문 11. 활성화 레코드(activation record)에 저장되지 않는 정보는?

① 지역 변수(local variable)

② 반환 주소(return address)

③ 동적 링크(dynamic link)

④ 전역 변수(global variable)

문 12. 명령형 언어(imperative language)에 대한 설명 중 옳지 않은 것은?

① 명령형 언어는 von Neumann 컴퓨터 구조에 영향을 받아서 설계된 언어이다.
② 명령형 언어에서의 변수는 von Neumann 컴퓨터 구조에서 메모리 셀을 형상화한 것이다.
③ Prolog나 LISP와 같은 언어들도 프로그램들이 명령어들로 이루어져 있기 때문에 명령형 언어라 할 수 있다.
④ C나 Pascal과 같은 언어들은 명령형 언어이기 때문에 Pentium과 같은 CPU의 기계어로 컴파일하기 쉽다.

문 13. Java 언어의 추상 클래스(abstract class)에 대한 설명으로 옳은 것은?

① 추상클래스로부터 객체를 생성할 수 있다.
② 추상클래스 형의 변수에 그 서브클래스 객체를 저장하는 것이 가능하다.
③ 추상클래스는 메소드의 구현부분을 가질 수 없다.
④ 추상클래스란 인터페이스와 같은 개념이다.

문 14. 다음 문법에 의해서 생성되는 문장은?

```
〈S〉→〈A〉ab〈B〉b
〈A〉→〈A〉a | b
〈B〉→b〈B〉 | a
```

① baabbab
② bababab
③ bbabbab
④ babbaab

문 15. Java 언어의 자료형에 대한 설명으로 옳지 않은 것은?

① Java 언어의 자료형은 크게 기본형과 참조형(reference type)으로 구분되며 배열(array), 클래스(class), 인터페이스(interface)는 참조형에 속한다.
② Java 언어의 정수형에는 byte(8비트), short(16비트), int(32비트), long(64비트)형이 지원되며 그 크기가 항상 고정되어 있다.
③ Java 언어의 배열은 new 연산자를 통해 정적으로 생성되며 스택 공간에 할당된다.
④ 문자형은 16비트 크기를 갖는 유니코드(unicode)를 사용한다.

문 16. C와 C++ 언어의 포인터형(pointer type)에 대한 설명으로 옳지 않은 것은?

① 포인터 변수는 힙-동적(heap-dynamic) 변수에 대한 접근을 제공할 수 있다.
② 허상 포인터(dangling pointer)는 할당된 기억장소가 이미 회수된 힙-동적(heap-dynamic) 변수에 대한 참조를 포함한다.
③ '*' 연산자는 역참조 연산을 의미하며, '&' 연산자는 변수의 주소를 반환한다.
④ void* 타입으로 정의된 변수는 nil 값을 가지며 가비지 포인터(garbage pointer)를 생성한다.

문 17. 다음 프로그램 구문의 계산 결과는?

```
5-2-1*2-1*3
```

〔단, 일반적인 프로그래밍 언어와는 달리, 우선순위(precedence)는 -가 *보다 높고, 결합순서(associativity)는 -, *모두 우측 우선(right associative)이다〕

① -2
② -1
③ 6
④ 12

문 18. 쓰레기 수집(garbage collection)에 대한 설명 중 옳지 않은 것은?

① 할당되어 있지만 더 이상 사용되지 않는 메모리 영역을 회수하여 다시 사용할 수 있도록 해준다.
② Java의 경우 자동 실시된다.
③ 블록 구조를 빠져나갈 때, 블록 내의 지역 변수용으로 할당된 메모리 영역을 회수한다.
④ 힙(heap) 메모리 관리에서 사용된다.

문 19. a+b*c가 항상 a+(b*c)로 해석될 수 있도록 하는 문법으로 가장 알맞은 것은?

① E ::= E+E | E*E | id
   id ::= a | b | c
② E ::= E+F | F
   F ::= F*id | id
   id ::= a | b | c
③ E ::= F*F
   F ::= E | F+id | id
   id ::= a | b | c
④ E ::= E+E | id+id | id
   id ::= a | b | c

문 20. 공유 데이터에 대한 접근을 제어하기 위한 모니터(monitor)에 대한 설명으로 옳지 않은 것은?

① 모니터는 동기화를 위해 P, V 연산을 제공한다.
② 모니터는 프로세스 간에 통신을 이용한 동기화 기능을 제공한다.
③ 모니터는 공유 데이터에 대한 프로세스들의 상호배타적 접근을 보장한다.
④ 모니터는 추상 자료형(abstract data type)을 지원하는 동기화 기법이다.