**R 데이터 구조와 알고리즘**

효율적인 데이터 구조와 알고리즘으로 애플리케이션의 속도와 성능을 높이자.

**저자 소개**

PKS 프라카시(PKS Prakash)

미국 위스콘신 메디슨 대학에서 산업 및 시스템 엔지니어링으로 박사학위를 받았다. 그리고 영국 워릭 대학교에서 두번째 공학박사 학위를 받았다. 그는 헬스케어, 제조업, 제약업, 그리고 전자상거래 분야 등 다양한 영역의 선두 기업에서 비즈니스와 관련된 예측 모델링, 가상 계측, 예방적인 유지보수, 근본 원인 분석, 프로세스 시뮬레이션, 사기 탐지, 조기 경보 시스템 구축 등의 업무에 데이터 과학자로 일해왔다. 현재 그는 Dream11 이라는 기업에서 부사장 및 실무 책임자로 일하고 있다. Dream11은 세계에서 가장 큰 판타지 크리켓, 축구, 카바디 게임을 제공하는 기업이다. 그는 제조업과 헬스케어 분야에서 경영 효율성 향상을 위한 연구 및 관리, 소프트웨어 도구, 그리고 고급 알고리즘에 대한 글을 IEEE-Trans, EJOR, IJPR 같은 주요 저널에 폭넓게 기고했다. “Evolutionary Computing in Advanced Manufacturing”의 한 장(chapter)을 쓴 공동저자이며, “Intelligent Approaches to Complex Systems”을 편집했다.

아슈튜니 스리 크리슈나 라오(Achyutuni Sri Krishna Rao)

싱가포르의 국립대학에서 기업 비즈니스 분석(데이터 과학)으로 석사 학위를 받았다. 그는 데이터 과학자로서 제조업, 헬스케어 및 제약업 분야에서 일했다. Rao는 R의 열렬한 지지자이며, 오픈 소스 커뮤니티에서 기여하기를 좋아한다. 그는 프리랜서로서, 기술 블로그(http://rcodeeasy.blogspot.com)로, 그리고 마라톤으로 그의 열정을 보여주고 있다. 현재 그는 선두적인 컨설팅 회사에서 데이터 과학 컨설턴트로 활동하고 있다.

감사의 말

이름을 일일이 열거할 수 없는 많은 사람들의 참여와 도움이 없었으면 이 책의 완성은 가능하지 않았을 것이다. 그들의 공헌에 진심으로 감사 드리며 고맙게 생각한다. 하지만 이 프로젝트에 참여한 Packt 출판의 모든 팀원들에게 가장 깊은 감사를 표시하고 싶다. 이 책은 Denim Pinto(원고 편집자)와 가진 초기 토론을 통해 아이디어가 생겼기 때문에 그에게 특별한 고마움을 전한다. 그가 없었다면 이 책은 절대로 나오지 않았을 것이다. 또한 이 책이 적절한 시기에 나오도록 힘써준 Pooja Mhapsekar와 Siddhi Chavan(컨텐츠 편집자), 그리고 Sunith Shetty(기술 편집자)에게도 감사를 표한다. 이 책의 품질이 좋아지도록 많은 피드백을 준 Vahid Mirjalili(감수자)에게도 감사의 말을 전하고 싶다.

감수자 소개

Vahid Mirjalili 박사는 소프트웨어 엔지니어/데이터 과학자로서 현재 미시건 주립대학교에서 컴퓨터 사이언스 박사 과정을 진행중이다. 통합 패턴인식과 생물측정 (Integrated Pattern Recognition and Biometrics, i-PRoBE) 연구소에서 그의 연구는 대규모 데이터셋에서 얼굴 이미지의 속성 분류에 관한 것이다. 게다가 그는 파이썬 프로그래밍 뿐만 아니라 데이터 분석 및 데이터베이스에 대한 컴퓨팅 개념을 가르치고 있다. 데이터 마이닝을 전공한 그는 예측 모델링과 데이터로부터 통찰력을 얻는 데 매우 관심이 많다. 그는 또한 파이썬 개발자로서 오픈 소스 커뮤니티에 기여하기를 좋아한다. 그는 데이터 과학과 컴퓨터 알고리즘의 다양한 영역에 대한 튜토리얼 만드는 것을 즐기며, 이 튜토리얼은 깃허브 저장소(<http://github.com/mirjalil/DataScience)>에서 확인할 수 있다.

<대> 서문

데이터 구조는 특정 데이터를 효율적으로 체계화하고 처리하는 방식을 나타낸다. 데이터 구조는 문제 해결에 매우 중요하며, 재사용 가능한 코드를 작성할 수 있는 완벽한 솔루션을 제공한다. “R 데이터 구조와 알고리즘”은 분석 및 인텔리젼스 분야에 일하고 있는 R 사용자들이 데이터 구조에 대한 역량을 강화하는 것에 목적을 두고 있다. R은 벨 연구소(예전에는 AT&T, 지금은 루슨트테크놀러지)에서 개발한 것으로 통계적인 처리와 시각화를 위해 매우 잘 설계된 언어이자 개발 환경이다. 이 책은 독자들이 알고리즘의 계산 효율성과 자원 사용 관점에서 최적화된 알고리즘을 설계할 수 있도록 해줄 것이다. 이 책은 여러가지 데이터 구조와 알고리즘과의 관계를 설명하고, 알고리즘 분석과 평가에 대해 얘기한 후, 알고리즘을 작성하는 프로세스로 나아갈 것이다. 고전적인 데이터 구조뿐만 아니라 함수적 데이터 구조의 통합성을 이해할 수 있도록 다룰 것이다. 리스트, 스택, 큐, 딕셔너리와 같은 데이터 구조의 기초와 함께 색인, 정렬, 검색까지 더 깊은 주제도 다룰 것이다. 또한 그래프, 동적 프로그래밍, 그리고 무작위 알고리즘과 같은 주제도 얘기할 것이다.

이 책의 목표는 R을 사용한 데이터 구조의 개념을 수립하는 데 있다.

<중> 이 책의 구성(## 이 부분은 에이콘에서 제목을 통일하기 때문에 수정합니다.)

1장, 시작하기. R 기초 수립에 중요한 데이터 구조 관련 배경 지식과 그 중요성을 이야기한다.

2장, 알고리즘 분석. 알고리즘 분석을 위한 동기 부여, 기본 표기법, 그리고 기초적인 기법에 대해 이야기한다.

3장, 링크드 리스트. 링크드 리스트의 기초를 세우고, 선형 링크드 리스트, 이중 링크드 리스트, 환형 링크드 리스트 등과 같은 링크드 리스트의 다양한 형태를 다룬다.

4장, 스택과 큐. 배열 기반의 그리고 링크드 리스트 기반의 스택과 큐를 소개하고 R에서 구현해본다.

5장, 정렬 알고리즘. 삽입 정렬, 버블 정렬, 선택 정렬, 쉘 정렬 등 다양한 정렬 알고리즘에 대해 설명하고, 서로 다른 알고리즘 간의 경험적 비교를 제공한다.

6장, 검색 옵션 탐색. 벡터 및 링크드 리스트를 포함한 리스트에 대한 검색 처리에 대해 상세히 알아본다. 또한 자기조직 리스트와 해시 개념도 소개한다.

7장, 색인. 디스크에서 파일을 구조화하고 대용량의 데이터를 체계화하는 데 핵심적인 색인 개념을 다룬다. ISAM, 2-3 트리, B-트리, and B+ 트리 등을 자세히 다룰 것이다.

8장, 그래프. 그래프 데이터 구조 및 구현을 위한 기초를 수립한다. 또한 횡단, 최단경로 문제, 최소 비용 신장 트리 알고리즘에 대해서도 알아본다.

9장, 프로그래밍과 무작위 알고리즘. 정적인 데이터 구조에서 무작위 스킵 리스트와 같은 무작위 데이터 구조로 개념을 확장한다. 그리고 이 장에서 프로그래밍 개념과 여러가지 애플리케이션을 소개한다.

10장, 함수적 데이터 구조. 함수적 데이터 구조와 지연 연산에 대해 소개한다. 그리고 R에서의 함수적 스택과 큐를 다룬다.

<중> 준비 사항(## 통일된 제목이 있어 수정합니다. 다른 Packt 책 번역 시 반영해주세요.)

알고리즘 설계와 데이터 과학에 대한 탐구심, 인내심, 그리고 열정이 필요하다. 데이터 구조가 다루는 범위와 그 적용은 매우 광범위하다.

R 또는 다른 프로그래밍 언어에 잘 알고 있으면 좋다. 프로그래밍과 데이터 분석에 대한 예비 경험도 도움이 될 것이다. 애플리케이션 개발에 상당히 도움이 되는 알고리즘에 대해 고마워하는 마음도 필요하다.

<중> 이 책의 대상 독자

이 책은 데이터 구조를 효율적으로 사용하고자 하고자 하는 R 개발자를 위한 것이다. R에 대한 기본 지식이 요구된다.

<중> 편집 규약

이 책은 다양한 정보를 구분하기 위해 여러가지 문장 스타일을 사용하고 있다. 여기서 그 스타일의 예와 의미를 설명한다.

문장 안에서 코드, 데이터베이스 테이블명, 폴더명, 파일명, 파일 확장자, 경로, 더미 URL, 사용자 입력값 등은 다음과 같이 표시한다. : “install.packages() 명령을 사용하여 R 콘솔에서 새로운 R 패키지를 설치하고 컴파일할 수 있게 해준다.”

코드 영역은 다음과 같이 표기한다.

if (테스트 표현식)

{

true일 경우 실행되는 명령문

}

명령창에 입력 또는 출력되는 내용은 다음과 같이 표기한다.

pip3 install --upgrade pip

pip3 install jupyter

새로운 용어와 중요한 단어는 보이는 것처럼 굵은 글씨로 강조했다. 메뉴, 대화창처럼 화면에 보이는 단어는 다음과 같이 나타낸다. : “새로운 R 노트북을 시작하려면 그림 1.7에 보이는 것처럼 오른쪽에 있는 New 탭을 클릭한 후 R kernel을 선택한다.”

[정보]

주의사항 또는 중요한 정보는 이와 같이 박스 안에 나타낸다.

[팁]

팁과 트릭은 이와 같이 나타낸다.

<중> 독자 의견

독자로부터의 피드백은 언제나 환영한다. 좋았는지 그렇지 않았는지 이 책에 대한 당신의 생각을 알려주기 바란다. 독자들의 피드백은 최대한 활용할 수 있는 타이틀을 개발하는데 도움이 되기 때문에 우리에게 정말 중요하다. 일반적인 피드백을 보낼 때는 [feedback@packtpub.com](mailto:feedback@packtpub.com) 으로 이메일 제목란에 책 제목을 넣어서 보내면 된다. 만약 당신이 특정한 주제에 대한 전문가이고 책을 쓰거나 공동 저술을 하는데 관심이 있다면 [www.packtpub.com/authors](http://www.packtpub.com/authors) 에서 저자 가이드를 보라.

<중> 고객 지원

당신은 이제 Packt 책의 자랑스런 소유자이므로, 이것을 최대한 활용할 수 있도록 우리는 많은 것을 제공하고 있다.

<소> 예제 코드 다운로드

이 책의 예제 코드는 <http://www.packtpub.com>에서 당신의 계정으로 로그인하여 다운로드 받을 수 있다. 이 책을 다른 곳에서 구매했다면 <http://www.packtpub.com/support>에 방문해서 가입하면 당신의 이메일로 소스코드를 받을 수 있다.

다음 과정을 통해 예제 코드를 다운로드 받을 수 있다.

1. 웹사이트에 이메일 주소와 패스워드를 사용해서 로그인 또는 가입한다.

2. 위쪽의 SUPPORT 탭을 선택한다.

3. Code Downloads & Errata를 클릭한다.

4. Search 부분에 책의 제목을 입력한다.

5. 원하는 책을 선택한다.

6. 책을 구매한 곳을 드롭다운 메뉴에서 선택한다.

7. Code Download를 클릭한다.

파일을 다운로드 받은 후 다음 프로그램의 최신 버전으로 압축을 해제하면 된다.

* 윈도우즈 : WinRAR / 7-Zip
* 맥 OS : Zipeg / iZip / UnRarX
* 리눅스 : 7-Zip / PeaZip for

이 책의 코드는 깃허브의 다음 저장소에서도 다운로드 받을 수 있다.

<https://github.com/PacktPublishing/R-Data-Structures-and-Algorithms>

그 외의 다양한 책과 비디오의 코드도 <https://github.com/PacktPublishing/> 에서 얻을 수 있다.

<소> 컬러 이미지 다운로드

스크린샷과 다이어그램 등의 컬러 이미지를 담고 있는 PDF 파일도 제공하고 있다. 컬러 이미지는 출력 결과의 변화를 더 잘 이해하는데 도움이 될 것이다. 다음 주소에서 다운로드 받을 수 있다.

<https://www.packtpub.com/sites/default/files/downloads/RDataStructuresandAlgorithms_ColorImages.pdf>

<소> 정오표

책의 내용이 정확하도록 모든 노력을 기울이고 있지만 실수는 일어나기 마련이다. 만약 책에서 문장이나 코드 등의 오류를 발견할 경우 우리에게 알려주면 매우 고마울 것이다. 그렇게 함으로써 당신은 다른 독자들을 혼란에서 구할 수 있으며, 이 책의 다음 버전을 향상시키는데 도움이 될 것이다.

어떤 오타라도 발견하게 되면 <http://www.packtpub.com/submit-errata>로 방문하여 책을 선택하고, Errata Submission Form 링크를 클릭한 후, 상세한 내용을 입력하면 된다. 당신의 정오표 제안이 검증되면 웹사이트에 업로드 되거나 기존의 정오표에 내용이 추가될 것이다.

<https://www.packtpub.com/books/content/support> 에 방문하여 검색창에 책 제목을 넣으면 이전에 제시된 정오표를 볼 수 있다. 요청된 정보가 Errata 섹션 아래에 나타난다.

<소> 저작권 침해

인터넷 상의 저작권 자료를 불법복제하는 것은 모든 미디어에서 문제가 되고 있다. Packt 출판사는 저작권과 라이선스 보호에 대해 매우 심각한 태도를 취하고 있다. 인터넷 상에서 어떤 형태로든 불법복제물을 발견하게 되면 우리가 즉각 조치를 취할 수 있도록 웹사이트 주소나 이름을 알려주기 바란다.

불법복제물로 의심되면 [copyright@packtpub.com](mailto:copyright@packtpub.com)으로 연락하면 된다. 저자와 귀중한 컨텐츠를 제공할 수 있는 능력을 보호하는데 도움을 준 여러분께 감사한다.

<소> 질문

만약 이 책의 내용에 대해 질문이 있다면 [questions@packtpub.com](mailto:questions@packtpub.com) 으로 문의하라. 최선을 다해 답변을 하도록 하겠다.

1장. 시작하기

빠르고 효율적인 정보 검색은 대다수 컴퓨터 프로그램의 일차적인 목적이다. 데이터 구조와 알고리즘은 데이터를 더 빠르게 처리하고 검색하려는 목적을 이루는 데 도움이 된다. 데이터로부터 추론될 수 있는 다음과 같은 질문에 대답할 때 정보 검색은 알고리즘과 쉽게 통합될 수 있다.

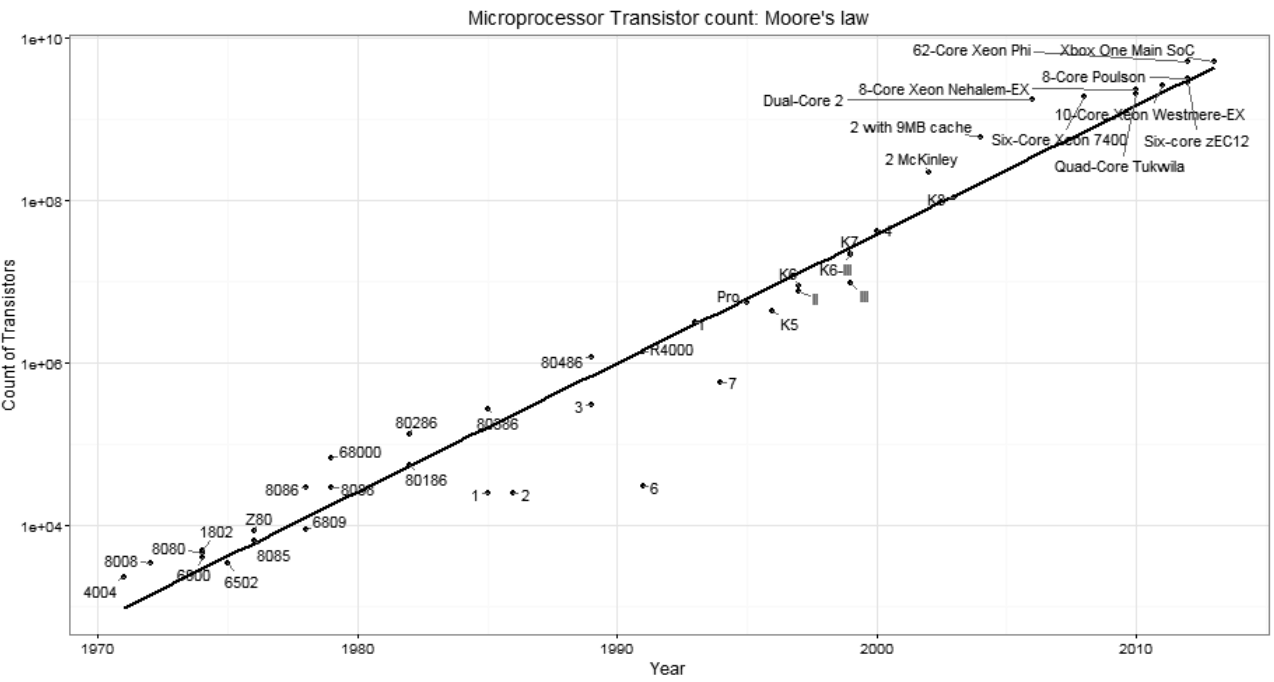
* 매출은 시간이 지남에 따라 어떻게 증가하는가?
* 고객의 방문 시간 분포는 어떠한가?
* 오후 3시에서 6시 사이에 방문한 모든 고객 중에서 아시아인 대비 중국인은 얼마나 많이 구매하는가?
* 모든 방문 고객 중에서 같은 도시에서 온 사람은 얼마나 되는가?

이 질문들에 대한 데이터를 처리하는 데 있어서, 특히 빅데이터 상황이라면, 데이터 구조와 알고리즘은 데이터 검색을 수행하는데 아주 중요한 역할을 한다. 이 책에서는 정보 처리에 일반적으로 사용되는 리스트, 큐, 스택과 같은 기본적인 데이터 구조와 그에 대립되는 다른 데이터 구조들에 대해 설명할 것이다. 또한 정의된 데이터 구조의 검색 및 처리 성능에 대한 데이터 구조 및 알고리즘 평가 방법에 대해서도 알아볼 것이다.

알고리즘은 복잡성과 효율성에 기반해 평가된다. 복잡성은 알고리즘 설계가 프로그래밍과 디버깅하기 쉽게 되어 있는지를 나타내고, 효율성은 알고리즘이 컴퓨터의 자원을 최적으로 사용하는지를 말해준다. 이 책은 데이터 구조를 사용하는 알고리즘의 효율성 부분에 초점을 맞출 것이며, 이 장에서는 데이터를 추출해내기 위해 사용되는 데이터 구조와 알고리즘의 중요성에 대해 이야기할 것이다.

<대> 데이터 구조 소개

집적회로(IC)는 발명된 이후로 평방 인치당 트랜지스터의 수가 매년 두 배씩 증가한다는, 1965년에 발표된 무어의 법칙에 따라 컴퓨터의 계산 능력은 향상되고 있다. 1975년 그는 포화 상태로 인해 매년이 아닌 2년마다 두 배가 된다고 예측을 수정했다.



<그림시작>

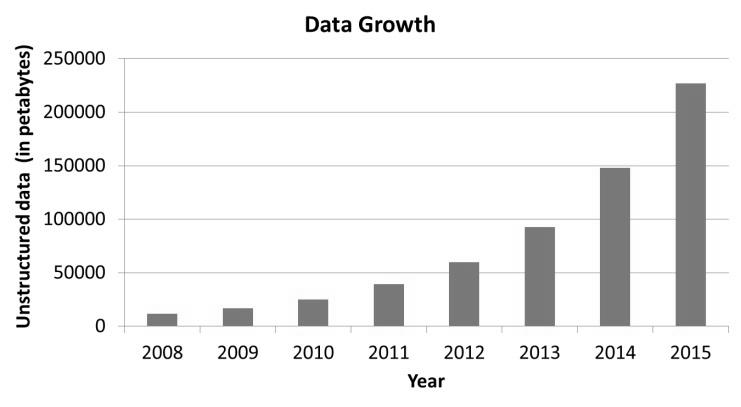
y축 : 트랜지스터의 갯수

x축 : 년도

<그림끝>

(##그림보다 아래에 위치하므로 <그림끝> 다음에 삽입하였습니다.)

컴퓨팅 능력이 계속 향상되고 있음에도 불구하고, 문제의 복잡성과 데이터 소스도 지난 10여 년간 기하급수적으로 증가하면서 효율적인 알고리즘의 필요성도 더 강조되고 있다.



<그림시작>

제목 : 데이터의 증가

y축 : 비정형 데이터 (페타바이트)

x축 : 년도

<그림끝>

2008년에서 2015년 사이에 일어난 데이터의 폭발은 정형, 반정형, 그리고 비정형 등 모든 종류의 데이터셋을 사용하여 통찰력을 끌어내기 위해 많은 노력을 기울이는 데이터 과학이라는 새로운 분야로 이끌었다. 그러므로 대규모의 데이터를 효율적으로 다루기 위해서는 데이터셋을 효율적으로 저장하고 검색하는 것이 매우 중요하다. 예를 들어, 사전에서 한 단어를 찾을 때 만약 데이터가 무작위로 구성되어 있다면 많은 시간이 걸릴 것이다. 그렇기 때문에 정렬된 리스트 데이터 구조는 단어의 빠른 검색을 보증한다. 또한 입력 위치를 기반으로 한 도시에서 최적의 이동 경로를 찾는 것은 도로 연결망, 위치 정보, 그리고 지오메트리 형태로 저장된 데이터를 필요로 한다. 이상적으로 문자, 정수, 부동소수 등과 같은 기존의 내장 데이터 타입으로 저장된 변수도 스칼라 유형의 데이터 구조라고 할 수 있다. 그러나 공식적으로 데이터 구조는 컴퓨터에서 연관된 정보를 조직화하여 효율적으로 사용할 수 있는 하나의 스킴(scheme)으로 정의된다.

알고리즘의 경우 충분한 공간과 시간이 주어진다면 관심있는 질문에 대답하기 위해 어떤 데이터셋이라도 저장하고 처리할 수 있다. 하지만 정확한 데이터 구조를 선택하는 것은 컴퓨터의 메모리와 자원을 절약하는데 상당한 도움이 된다. 예를 들어, 매일 방문하는 고객의 수를 정수형 데이터 타입이 아닌 부동소수형 데이터 타입으로 설정한다면 두 배의 메모리를 필요로 하게 된다. 하지만 현실세계에서 컴퓨터의 자원과 공간은 항상 제한되어 있다. 그러므로 주어진 자원과 시간 안에서 원하는 목표를 이룰 수 있다면 효과적인 솔루션이라고 할 수 있다. 이것은 알고리즘을 설계하는 동안 서로 다른 데이터 구조들 간의 성능을 비교하는 비용함수로 사용할 수 있다. 데이터 구조를 선택할 때 고려해야 할 두 가지 주요한 제약은 다음과 같다.

* 선택한 데이터 구조에서 지원해야 하는 항목 추가, 항목 삭제, 검색과 같은 기본적인 작업을 결정하기 위해 문제를 분석
* 각 작업에 대한 자원 제약사항을 평가

데이터 구조는 문제 상황에 따라 선택된다. 예를 들어, 전체 데이터가 초기에 로드되고, 데이터에 대한 변경이나 추가가 없으면 비슷한 데이터 구조가 요구된다. 하지만 위 상황에서 데이터 구조에 삭제 작업이 포함된다면 데이터 구조 구현은 좀더 복잡해질 것이다.

[팁]

코드를 다운받는 자세한 방법은 이 책의 서문에 있으니 꼭 확인하기 바란다. 이 책을 위한 코드는 깃허브의 다음 주소에 있다. <https://github.com/PacktPublishing/R-Data-Structures-and-Algorithms>. 또한 출판사의 다양한 책들과 비디오의 소스 코드는 다음 주소에서 확인 가능하다.

<https://github.com/PacktPublishing/>

<대> 추상 데이터 타입과 데이터 구조

추상 데이터 타입(abstract data type, ADT)은 데이터 구조에 대한 기능과 처리를 고수준에서 정의할 때 사용되며, 데이터 구조를 상세하게 구현하기 전에 확인해야 한다. 예를 들어, 링크드 리스트를 구현하기 전에 정의된 링크드 리스트에서 다음 중 수행하려는 작업이 무엇인지 아는 것이 좋을 것이다.

* 링크드 리스트에 항목을 추가할 수 있어야 한다.
* 링크드 리스트로부터 항목을 삭제할 수 있어야 한다.
* 링크드 리스트에서 항목을 검색할 수 있어야 한다.
* 링크드 리스트가 비어 있는지 아닌지 확인할 수 있어야 한다.

정의된 추상 데이터 타입은 전략을 수립하기 위해 필요하다. 이 책에서는 여러가지 데이터 구조에 대한 추상 데이터 타입을 더 자세하게 알아볼 것이다. 추상 데이터 타입의 정의를 말하기 전에, 데이터 구조를 위한 생태계를 구성하는 데이터 타입과 그 특성에 대해 먼저 이해하는 것은 매우 중요하다.

데이터 타입은 불리언(boolean), 정수(integer), 부동소수(float), 문자열(string) 등과 같이 데이터의 다양한 유형을 분류하는 방법이다. 데이터셋을 효율적으로 분류하기 위해서 모든 데이터 타입은 다음과 같은 특성을 가져야 한다.

* 원자성(atomic) : 하나의 단위 개념으로 정의될 수 있어야 한다.
* 추적 가능성(traceable) : 동일한 데이터 타입으로 묶을 수 있어야 한다.
* 정확성(accurate) : 모호하지 않아야 한다.
* 명확성과 간결성(clear and concise) : 이해 가능해야 한다.

데이터 타입은 다음과 같이 두 가지로 나눌 수도 있다.

* 내장(Built-in) 데이터 타입
* 파생(Derived) 데이터 타입

한 언어에 내장되어 지원되는 데이터 타입을 내장 데이터 타입이라고 한다. R은 다음과 같은 데이터 타입을 지원한다.

* 정수형(Integers)
* 부동소수형(Float)
* 불리언(Boolean)
* 문자열(Character)

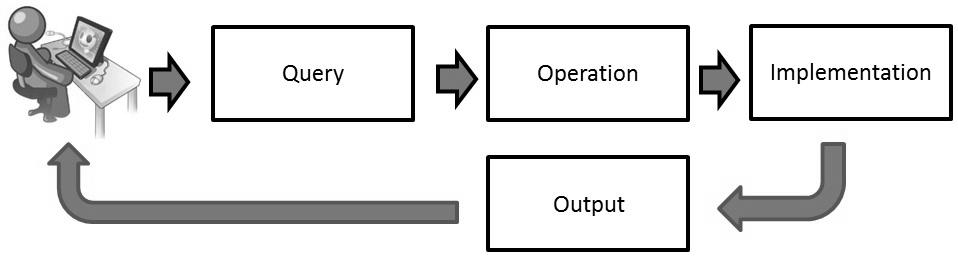
내장 데이터 타입과 통합되어 추가, 삭제, 정렬, 병합 등과 같은 연관된 작업을 처리하기 위해 다음과 같은 파생 데이터 타입이 있다.

* 리스트(List)
* 배열(Array)
* 스택(Stack)
* 큐(Queue)

파생 데이터 타입 또는 데이터 구조는 두 가지 부분에서 연구된다.

* 추상 데이터 타입 또는 수학/논리 모델
* 프로그램 구현

추상 데이터 타입은 소프트웨어에서 데이터 타입의 실현이다. 우리는 일반적으로 데이터 구조에 대해 사용자가 사용하는 고수준의 기능이나 작업에 관심을 갖지만, 내부적으로 이 기능들이 어떻게 동작하는지에 대해서는 모른다. 예를 들어, 한 사용자가 금융 소프트웨어에서 검색기능을 사용하여 스미스 씨의 거래 이력을 검색한다고 하자. 사용자는 이 작업이 동작하는 방법 또는 데이터 구조의 자세한 구현에 대해서는 전혀 알지 못한다. 그러므로 추상 데이터 타입의 동작은 오직 입력과 출력에 의해서만 관리된다.



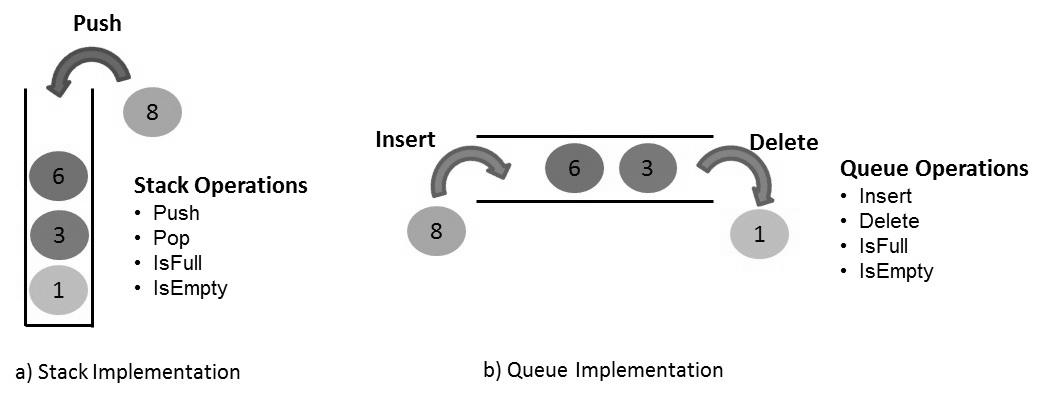
<그림시작>

질의 처리 구현

출력

<그림끝>

추상 데이터 타입은 데이터 타입이 어떻게 구현되는지 알 수 없다. 사용자에게 숨겨지고 외부의 접근으로부터 보호되기 때문인데, 이것이 캡슐화의 개념이다. 데이터 구조는 추상 데이터 타입이 프로그래밍 언어에 의해 구현되는 부분이다. 추상 데이터 타입은 그림 1.4에 보이는 것처럼 여러가지 구현 전략에 의해 달성될 수 있다.



<그림시작>

a) 스택 구현

b) 큐 구현

<그림끝>

추상 데이터 타입에서 제공되는 추상화는 프로그래밍의 복잡성을 관리하는데 도움이 된다. 추상 데이터 타입은 구현에 필요한 형식과 작업을 결정하기 때문에 논리 형식이라고 한다. 추상 데이터 타입은 특정 형식의 데이터 구조를 사용하여 구현된다. 추상 데이터 타입을 구현하기 위해 사용한 데이터 구조는 데이터 타입의 물리적인 형식이다.

<대> 문제와 알고리즘과의 관계

문제는 수행되어야 할 작업으로 정의할 수 있다. 수행할 작업의 실행은 일차적으로 두 요소로 나눌 수 있다.

* 데이터
* 알고리즘

그러나 문제의 제약사항, 자원 제약사항, 허용된 시간 등 개발에 영향을 줄 수 있는 다른 관리 요인이 있을 수도 있다.

문제의 데이터 요소는 숫자, 텍스트, 파일과 같이 우리가 다루어야 할 정보를 나타낸다. 예를 들어, 회사의 직원 정보를 관리하길 원한다고 가정하면 거기에는 직원 이름 및 그와 관계된 자세한 항목들이 포함된다. 이 데이터는 정기적으로 관리되고 업데이트 되어야 한다.

문제의 알고리즘 부분은 상세한 구현을 나타낸다. 여기에는 현재의 데이터를 어떻게 관리할 것인가 하는 문제가 수반된다. 데이터와 문제의 요구사항에 따라 데이터 구조를 선택한다. 그리고 데이터 구조에 따라 데이터셋을 관리하기 위한 알고리즘을 정의해야 한다. 예를 들어, 회사의 직원 데이터셋을 링크드 리스트 또는 딕셔너리에 저장할 수 있다. 데이터를 저장하기 위해 정의된 데이터 구조를 기반으로 검색, 추가, 삭제가 이루어지며, 데이터 구조 상에 수행되는 작업들은 알고리즘에 의해 제어되고 프로그램으로서 구현된다. 그러므로 프로그램은 어떤 작업을 하기 위해 컴퓨터에 주어지는 단계적인 명령이라고 할 수 있다.

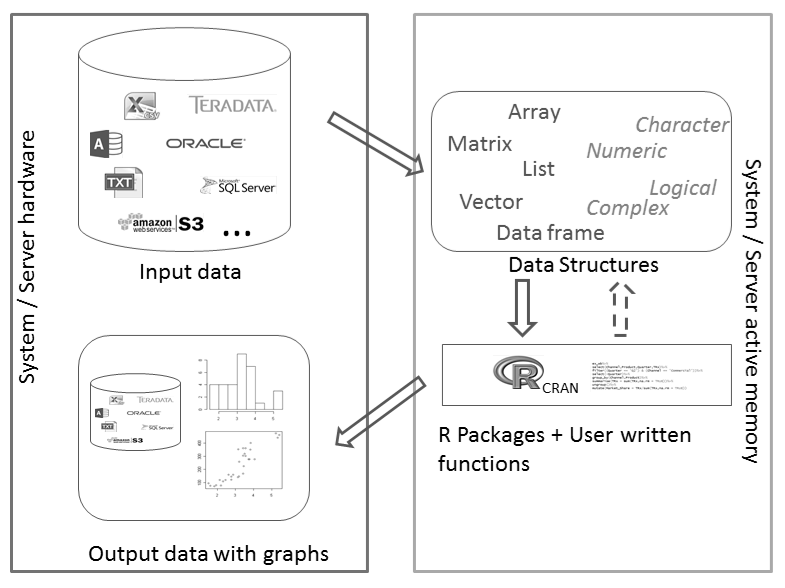
프로그램 = f(알고리즘, 데이터 구조)

정리하면, 프로그램은 모든 문제와 자원 제약사항을 고려하여 선택된 알고리즘을 사용하여 정의된 문제를 해결하기 위한 단계적인 명령들의 그룹이다. 이 책에서는 여러가지 데이터 구조와 알고리즘의 구현을 시연하기 위해 R 을 사용할 것이다.

<대> R 기초

R은 Ross Ihaka와 Robert Gentleman에 의해 설계되고 만들어진 통계 프로그래밍 언어이다. 이것은 AT&T의 벨 연구소에서 만든 S 언어에서 파생되어 나왔다. 통계 분석뿐만 아니라, R은 강력한 시각화 기능을 지원한다. R은 오픈 소스이며, 일반 공중 라이선스(GPL) 하에서 자유롭게 배포 가능하다. Comprehensive R Archive Network(CRAN)이라는 저장소에는 다양한 분석에 사용되는 8,400 개 이상의 패키지가 있으며, 무료로 설치 및 사용할 수 있다.

R은 인터프린터 기반의 직관적인 문법을 사용하는 고수준의 언어이다. R은 시스템 또는 서버의 메모리에서 실행되며, 실행 환경 내의 모든 파일, 함수, 그리고 파생된 결과들은 모두 객체로 저장된다. R이 실행되는 아키텍처는 그림 1.5에서 볼 수 있다.



<그림시작>

(왼쪽) 시스템/서버 하드웨어

입력 데이터

출력 데이터, 그래프

(오른쪽) 시스템/서버 활성 메모리

데이터 구조

R 패키지 + 사용자 정의 함수

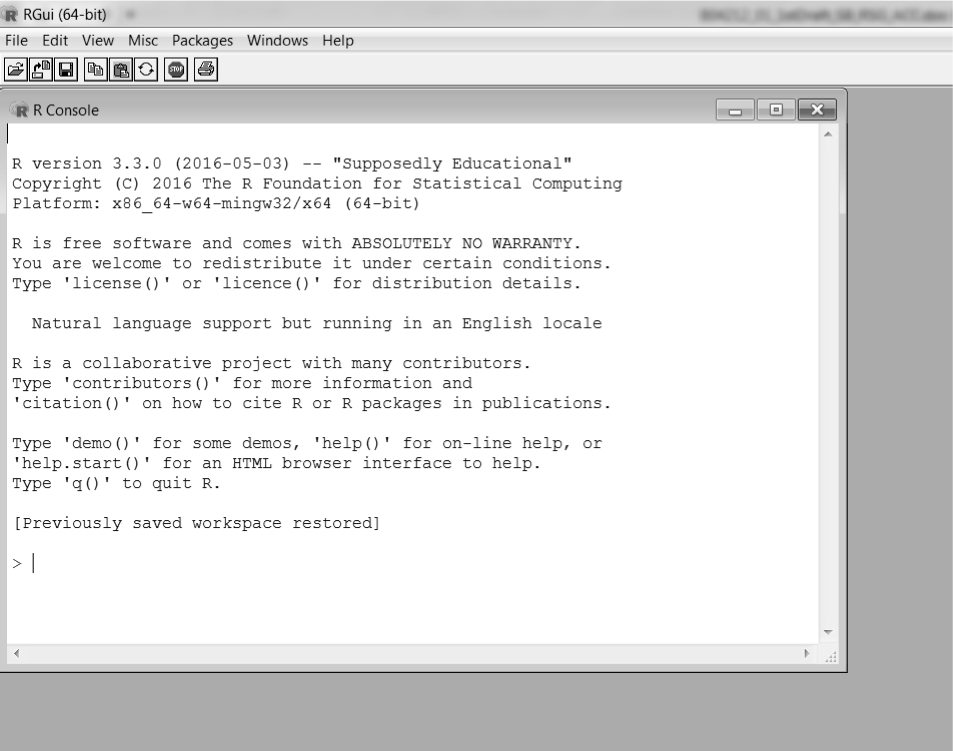
<그림끝>

**<중> R 설치**

R은 윈도우, 맥 OS X, 그리고 리눅스 등 모든 OS에 설치된다. 최신 버전의 설치 파일은 CRAN (<https://cran.r-project.org>) 및 여러 미러 사이트에서 다운로드 받을 수 있다. 또한 32비트와 64비트 아키텍처를 모두 지원한다.

r-base-dev는 많은 내장 함수들을 가지고 있기 때문에 설치를 권장한다. 이는 또한 install.packages() 명령을 사용하여 R 콘솔에서 직접 새로운 R 패키지를 설치하고 컴파일할 수 있게 해준다.

설치 후에 R은 프로그램 파일, 바탕화면 단축 아이콘, 또는 명령어 입력창 등을 통해 호출할 수 있다. 기본 설정 상태에서 R 콘솔은 다음과 같이 보인다.



<그림 1.6 : 코딩과 결과 확인을 즉시 시작할 수 있는 R 콘솔>

R은 주피터 노트북(Jupyter Notebook) 안에서 커널로 사용될 수 있다. 주피터 노트북은 문서 작성, 코드 작성과 결과 확인을 통합적으로 할 수 있게 해주는 웹 기반 애플리케이션이다. (명령창에서) pip 명령을 사용하여 주피터 노트북을 설치할 수 있다.

pip3 install --upgrade pip

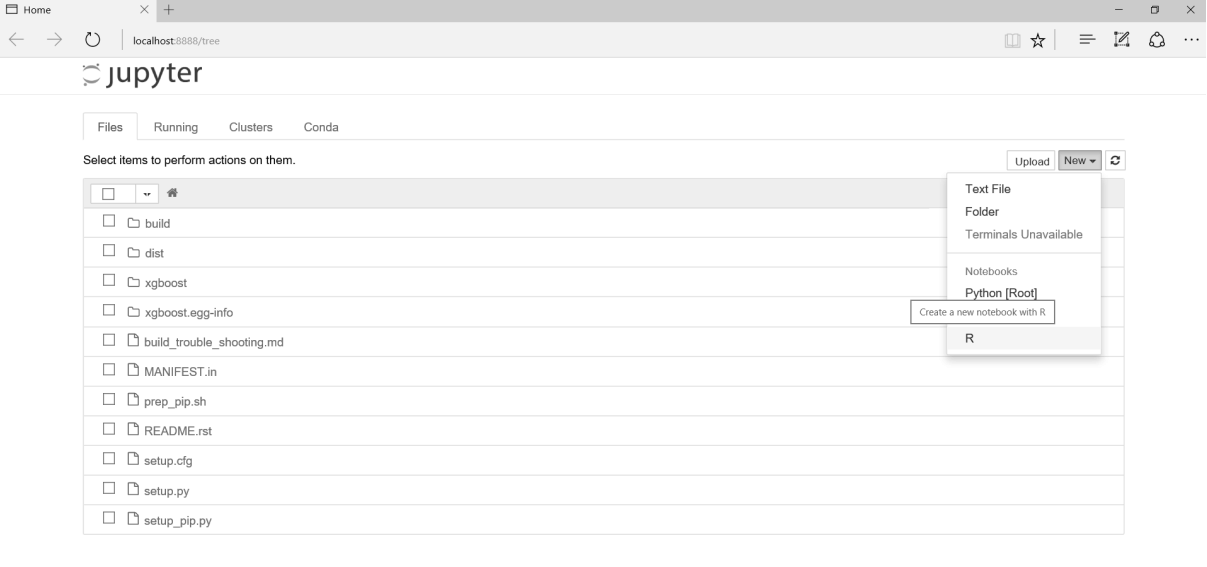
pip3 install jupyter

쉘 또는 명령창을 열고 다음 명령을 실행하면 브라우저에 주피터 노트북 인터페이스가 시작된다.

jupyter notebook

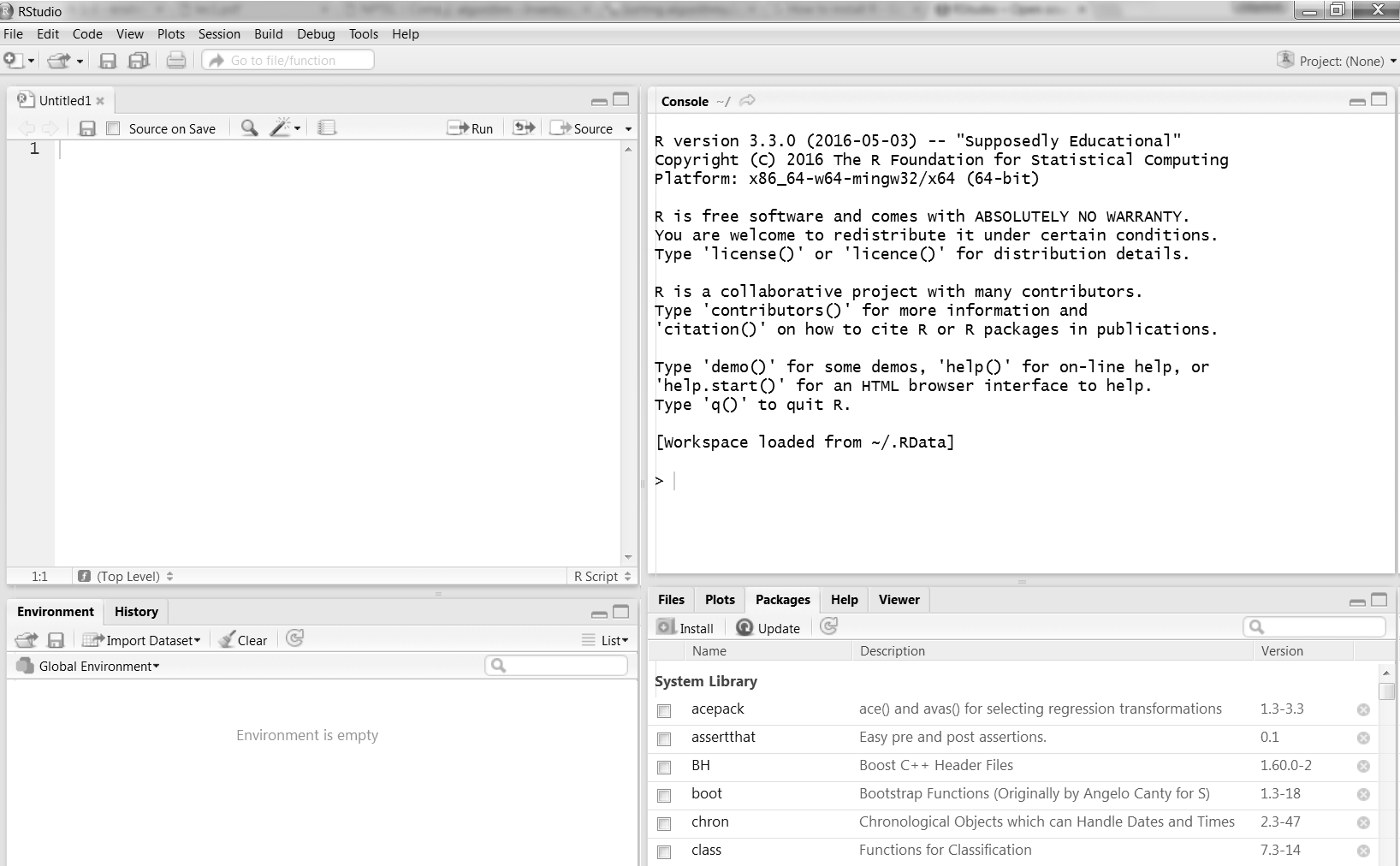
R 노트북을 시작하려면, 오른쪽의 New 탭을 클릭하고, 그림 1.7에 보이는 것처럼 R 커널을 선택한다. R 커널은 파이썬처럼 주피터 노트북의 기본 사양이 아니다. 파이썬과 주피터 설치시에 권장되는 아나콘다(Anaconda) 배포판은 <https://www.continuum.io/downloads>에서 다운로드 받을 수 있다. R essentials는 (아나콘다 설치 후에) 다음 명령을 통해 설치할 수 있다.

conda install -c r r-essentials



<그림 1.7: R 노트북 생성을 위한 주피터 노트북>

노트북이 생성되면 각 셀에서 코드 작성을 시작할 수 있다. R은 공식적인 컴파일을 필요로 하지 않고 런타임에 코드를 실행하기 때문에, 코딩을 하면 바로 결과를 확인할 수 있다. 콘솔 화면은 Windows 탭 아래의 몇 가지 옵션을 사용하여 조정할 수 있다. 하지만 R을 위한 강력하고 생산적인 사용자 인터페이스를 제공하는 통합개발환경(Integrated Development Environment, IDE)을 사용할 것을 강력히 권한다. 널리 사용되는 통합개발환경 중 하나로 무료이며 오픈 소스인 RStudio가 있다. 자체 서버인 RStudio Server Pro도 제공된다. Rstudio의 인터페이스는 아래 보이는 스크린 샷과 같다.



<그림 1.8: R을 위해 널리 사용되는 통합개발환경, RStudio >

<중> R의 기본 데이터 구조

(##역자: ‘데이터 타입’과 데이터 구조’가 혼용되고 있긴 하지만, 아래 표 제목은 ‘데이터 구조’로 되어 있으며 그 밑의 내용으로 봐서 중제목은 ‘데이터 구조’로 하는 것이 좋을 듯 합니다.)

R은 다양한 데이터 구조을 지원하며, 표 1.1과 같이 차원과 각 요소의 타입(동질성 또는 이질성)에 따라 분류할 수 있다.

|  |  |  |
| --- | --- | --- |
|  | 동질성 | 이질성 |
| 1 차원 | 개별 벡터 | 리스트\* |
| 2 차원 | 매트릭스 | 데이터 프레임 |
| n 차원 | 배열 |  |

\* 리스트는 복합적인 용도로 인해 n 차원으로 변환될 수 있다.

<표 1.1 R의 기본 데이터 구조>

동질성 데이터 구조는 모두 같은 타입의 컨텐츠로 구성되며, 이질성 데이터 구조는 다양한 타입의 컨텐츠를 가질 수 있다. 데이터 테이블 등의 다른 모든 데이터 구조는 기본 데이터 구조에서 파생될 수 있다. 데이터 타입과 특성에 대해서는 3장 링크드 리스트에서 자세히 다룰 것이다.

<중> R의 연산자

R에서 연산자(operator) 문법은 다른 프로그래밍 언어와 매우 비슷하다. 다음은 연산자의 목록과 그에 대한 설명이다.

다음 표는 다양한 산술 연산자이다.

|  |  |
| --- | --- |
| 연산자 | 설명 |
| + | 더하기 |
| - | 빼기 |
| \* | 곱하기 |
| / | 나누기 |
| \*\* 또는 ^ | 제곱 |
| %% | 나눈 나머지 |
| %/% | 나눈 정수의 몫 |

<표 1.2 R의 기본 산술 연산자>

다음 표는 다양한 논리 연산자이다.

|  |  |
| --- | --- |
| 연산자 | 설명 |
| == | 정확히 같다 |
| < | ~ 보다 작다 |
| > | ~ 보다 크다 |
| <= | ~ 보다 작거나 같다 |
| >= | ~ 보다 크거나 같다 |

<표 1.3 R의 기본 논리 연산자>

R에서 실행해볼 수 있는 아래 예제를 살펴보자. 우선 V라는 벡터를 할당한 후 이 벡터에 더하기, 빼기, 제곱근, 그리고 로그와 같은 연산을 수행했다. 어떤 연산이든 벡터는 각각의 요소에 적용된다.

> V <- c(1,2,3,4,5,6) ## 벡터 V를 할당한다.

> V

[1] 1 2 3 4 5 6

> V+10 ## 벡터 V의 각 요소에 10을 더한다

[1] 11 12 13 14 15 16

> V-1 ## 벡터 V의 각 요소에서 1을 뺀다

[1] 0 1 2 3 4 5

> sqrt(V) ## 벡터 V의 각 요소에 제곱근 연산을 수행한다

[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490

> V1 <- log(V) ## 벡터 V에 로그 변환을 한 후 V1에 저장한다

> V1

[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595

<중> R의 제어문

루프(loop)와 조건식 같은 제어문은 모든 프로그래밍 언어의 필수적인 부분이며, R은 아주 직관적인 문법으로 제어문을 지원한다.

<소> If 조건문

if 조건문의 문법은 다음과 같다.

if (테스트 표현식)

{

조건이 true일 경우 실행할 명령문

}

테스트 표현식이 true이면 명령문을 실행한다. 표현식이 false이면 아무 일도 일어나지 않는다.

다음 예제는 벡터 x를 정의한 후, x가 5보다 큰지 작은지 확인하기 위해 두 개의 조건문으로 분리하여 넘겼다. 두 조건 중 하나라도 만족되면 x 값은 콘솔에서 해당 if문의 바로 다음 줄에 출력된다.

> x <- 10

> if (x < 5) print(x)

> if (x > 5) print(x)

[1] 10

<소> If…else 조건문

if … else 조건문의 문법의 다음과 같다.

if (테스트 표현식)

{

true일 경우 실행되는 명령문

} else {

false일 경우 실행되는 명령문

}

이 시나리오는 테스트 표현식이 true이면 if 조건 아래에 있는 명령문이 실행되고, 그렇지 않으면 else 문 아래의 명령문이 실행된다. 다음 예제에서 10을 가진 벡터 x를 정의한다. 그런 다음 x를 2로 나누었을 때 그 나머지로 1을 반환하는지 (R은 1을 불리언 값 true으로 읽는다) 확인한다. True이면 x값과 함께 홀수라고 출력하며, 그렇지 않으면 짝수라고 출력한다.

> x=10

> if (x %% 2)

+ {

+ print(paste0(x, " :홀수"))

+ } else {

+ print(paste0(x, " :짝수"))

+ }

[1] "10 :짝수"

\*역자주 : RStudio에서 한글을 사용할 경우에는 텍스트 인코딩을 UTF-8로 설정한 후 사용할 것을 권장한다. 위와 같이 R 콘솔에서 한글을 사용할 경우 출력 결과에 한글이 정확히 표현되지 않을 수도 있으니 가능한 Rstudio 사용을 권장한다.

<소> Ifelse 함수

ifelse() 함수는 일차적으로 벡터에 사용하는 조건문이며, if...else 조건문과 동일한 형식이다. 이 조건부 함수는 벡터의 각 요소에 개별적으로 적용된다. 따라서 이 함수의 입력값도 벡터이고, 출력값도 역시 벡터이다. ifelse() 함수의 문법은 다음과 같다.

ifelse (테스트 표현식, true일 경우 실행할 명령문, false일 경우 실행할 명령문)

이 함수의 첫번째 인수는 테스트 표현식이며, 반드시 논리적인 결과를 반환해야 한다. 조건이 true이면 바로 다음의 명령문을 실행하며, 그렇지 않으면 마지막의 명령문을 실행한다.

다음 예제에서는 벡터 x에 1부터 6까지 정수값을 할당한다. 벡터 x의 각 요소에 조건을 적용하여 각 요소가 홀수인지 짝수인지 확인한다.

(##역자: 1 to 7 은 오류)

> x <- 1:6

> ifelse(x %% 2, paste0(x, " :홀수"), paste0(x, " :짝수"))

[1] "1 :홀수" "2 :짝수" "3 :홀수"

[4] "4 :짝수" "5 :홀수" "6 :짝수"

<소> For 루프

for 루프는 기본적으로 연속된 값들로 이루어진 벡터에 대해 명령문을 반복할 때 사용된다. 벡터는 숫자, 문자, 불리언 또는 복합형 등 어떤 타입이라도 가능하다. 반복할 때마다 명령문이 수행된다. for 루프의 문법은 다음과 같다.

for(x in 연속형 벡터) {

반복 수행될 명령문

}

루프는 벡터 안에 있는 모든 요소가 모두 소모될 때까지 계속된다.

다음은 for 루프의 예제이다. 벡터 x에 값을 할당하고, for 루프를 통해 x의 각 요소를 콘솔에 출력한다.

> x <- c("John", "Mary", "Paul", "Victoria")

> for (i in seq(x)) {

+ print(x[i])

+ }

[1] "John"

[1] "Mary"

[1] "Paul"

[1] "Victoria"

<소> 중첩 for 루프

중첩 for 루프(nested for loop)는 아래 보이는 것처럼 for 루프 안에 또다른 for 루프를 정의한 것이다.

for(x in 연속형 벡터)

{

반복 수행될 첫번째 명령문

for(y in 연속형 벡터)

{

반복 수행될 두번째 명령문

.........

}

}

중첩 for 루프에서, 이전 for 루프의 연속형 벡터를 기반으로 가능한 모든 횟수만큼 후속 for 루프가 반복 수행된다. 이것은 3×3 매트릭스를 mat에 정의하는 다음 예제를 통해 설명할 수 있다. 최종 목표는 연속된 더하기 연산을 통해 매트릭스 안에 있는 모든 값의 총합을 구하는 것이다. 첫번째로 sum 을 0으로 초기화한다. sum 변수는 순차적으로 매트릭스 내의 모든 요소와 자신을 더한 값으로 업데이트 된다. 매트릭스의 각 행의 모든 열의 값을 sum에 더하는 연산 순서로 중첩 for 루프로 정의했다.

> mat <- matrix(1:9, ncol = 3)

> sum <- 0

> for (i in seq(nrow(mat)))

+ {

+ for (j in seq(ncol(mat)))

+ {

+ sum <- sum + mat[i, j]

+ print(sum)

+ }

+ }

[1] 1

[1] 5

[1] 12

[1] 14

[1] 19

[1] 27

[1] 30

[1] 36

[1] 45

<소> While loop

R에서 while 루프는 특정 조건이 만족되다면 계속 반복하는 루프이다. 그 문법은 다음과 같다.

while (테스트 표현식)

{

조건이 true일 경우 수행할 명령문 (반복적 수행)

}

예제를 통해 while 루프를 자세히 알아보자. 객체 i를 1로 초기화한다. 매 반복마다 만족되어야 할 테스트 표현식은 i<10 이다. 처음에 i = 1이므로 조건이 true가 되어 while 루프 안의 명령문이 수행된다. 명령문에 의하면 i는 콘솔에 출력되고 그 다음에 1을 증가시킨다. 이제 i는 2가 되었고, 다시 한번 테스트 표현식의 조건이 true인지 false인지 확인한다. 그 결과가 true라면 명령문이 수행된다. 루프는 조건이 false가 될 때까지, 이 경우에는 i가 10이 될 때까지 반복된다. 여기서 i가 증가되는 부분은 매우 중요하며, 이것이 없으면 루프가 무한 반복될 수 있다.

> i <- 1

> while (i < 10)

+ {

+ print(i)

+ i <- i + 1

+ }

[1] 1

[1] 2

[1] 3

[1] 4

[1] 5

[1] 6

[1] 7

[1] 8

[1] 9

<소> 루프 내에서 사용하는 특수한 명령문

R에서는 break 또는 next 명령으로 루프를 변경할 수 있다. 이것은 명령 수행에 요구되는 다른 조건들을 루프 내에 추가하는데 도움이 된다.

**<최소> Break 명령**

break 명령문의 문법은 그냥 break이다. break는 남은 반복을 멈추고 루프를 종료할 때 사용된다. break 명령이 중첩 루프에서 사용될 경우 break가 언급된 가장 안쪽 루프는 종료되지만 그 바깥쪽 루프에는 영향을 미치지 않는다.

다음은 i가 8이 되면 루프가 종료되는 예제이다.

> for (i in 1:30)

+ {

+ if (i < 8)

+ {

+ print(paste0("Current value is ",i))

+ } else {

+ print(paste0("Current value is ",i," and the loop breaks"))

+ break

+ }

+ }

[1] "Current value is 1"

[1] "Current value is 2"

[1] "Current value is 3"

[1] "Current value is 4"

[1] "Current value is 5"

[1] "Current value is 6"

[1] "Current value is 7"

[1] "Current value is 8 and the loop breaks"

**<최소> Next 명령**

next 명령문의 문법도 그냥 next이다. next 명령은 조건에 기반하여 루프 내의 중간 과정을 반복하지 않고 넘길 때 사용된다. next 명령에 대한 조건이 만족되면 루프 내에서 그 아래의 모든 작업을 수행하지 않고 다음 순번의 반복을 시작한다.

아래는 2로 나누어서 나머지가 1인 조건을 만족하는 홀수인 경우만 출력하는 예제이다.

> for (i in 1:10)

+ {

+ if (i %% 2)

+ {

+ print(paste0(i, " : 홀수"))

+ } else {

+ next

+ }

+ }

[1] "1 : 홀수 "

[1] "3 : 홀수 "

[1] "5 : 홀수 "

[1] "7 : 홀수 "

[1] "9 : 홀수 "

<소> 반복 루프

반복 루프(repeat loop)는 어떤 조건(테스트 표현식)이 없이 수없이 반복되는 무한 루프의 일종이다. 그렇기 때문에 사용자는 break 명령문을 사용하여 루프가 종료되도록 반드시 종료 조건을 명확하게 언급해 주어야만 한다. 반복 루프의 문법은 다음과 같다.

repeat {

break 명령과 명확한 종료 조건을 가지고 있는 반복할 명령문

}

다음 예제는 i를 1로 초기화한다. 그 다음, 한 번의 for 루프 반복마다 i를 세 제곱한 값을 cube 객체에 넣고 729 보다 큰지 아닌지 조건을 검증한다. 동시에 i를 1만큼 증가시킨다. 조건이 만족되면 break 명령을 통해 for 루프가 종료된다.

> i <- 1

> repeat

+ {

+ cube <- i \*\* 3

+ i <- i + 1

+ if (cube < 729)

+ {

+ print(paste0(cube, " : 729 보다 작다. 루프에 머무른다."))

+ } else {

+ print(paste0(cube, " : 729 보다 크거나 같다. 루프에서 나온다."))

+ break

+ }

+ }

[1] "1 : 729 보다 작다. 루프에 머무른다."

[1] "8 : 729 보다 작다. 루프에 머무른다."

[1] "27 : 729 보다 작다. 루프에 머무른다."

[1] "64 : 729 보다 작다. 루프에 머무른다."

[1] "125 : 729 보다 작다. 루프에 머무른다."

[1] "216 : 729 보다 작다. 루프에 머무른다."

[1] "343 : 729 보다 작다. 루프에 머무른다."

[1] "512 : 729 보다 작다. 루프에 머무른다."

[1] "729 : 729 보다 크거나 같다. 루프에서 나온다 "

<대> R의 1등급 함수

R은 그 핵심이 기본적으로 함수적 언어이다. R에서 함수는 데이터 타입처럼 취급되며, 1등 시민으로 여겨진다. 다음 예제는 R이 모든 것을 함수 호출로 생각한다는 것을 잘 보여준다. 여기서 + 연산자 자체도 하나의 함수이다.

> 10+20

[1] 30

> "+"(10,20)

[1] 30

다음은 ^ 연산자도 함수임을 보여준다.

> 4^2

[1] 16

> "^"(4,2)

[1] 16

자, 이제 R 프로그래머에겐 매우 중요하고 또 널리 사용되는 함수라는 개념에 뛰어들어 보자.

벡터화된 함수는 프로그래머가 주어진 벡터의 각 개별 요소 수준에서 함수를 실행할 수 있도록 해주는 가장 유명한 함수의 개념 중 하나다. 이 벡터는 데이터 프레임, 매트릭스, 또는 리스트의 일부분 일수도 있다. 이해를 돕기 위해 다음 예제를 자세히 살펴보자. 값이 정의된 벡터 V\_in 내 각각의 요소에 연산을 수행하려고 한다. 연산은 벡터의 각 요소를 제곱한 후 V\_out 벡터로 출력하는 것이다. 다음과 같이 세 가지 접근법을 사용하여 구현할 수 있다.

접근법 1 : for 루프를 사용하여 각 요소 수준에서 연산을 수행한다. 이것은 S 언어 스타일을 사용하여 벡터 할당을 수행하기 때문에 세 가지 방법 중 가장 원시적인 방법이다.

> V\_in <- 1:100000 ## 입력 벡터

> V\_out <- c() ## 출력 벡터

> for(i in V\_in) ## 입력 벡터에 대한 for 루프

+ {

+ V\_out <- c(V\_out,i^2) ## 출력 벡터에 저장

+ }

접근법 2 : 여기서 벡터화된 함수 개념이 동일한 목적을 이루는 데 사용된다. 벡터화된 프로그래밍에서 루프는 (R 내부적으로) C 언어로 구현되어 있기 때문에 접근법 1처럼 R에서 구현되는 for 루프보다 훨씬 빠르다. 아래 작업을 실행하는 데 걸리는 시간은 순식간이다.

> V\_in <- 1:100000 ## 입력 벡터

> V\_out <- V\_in^2 ## 출력 벡터

접근법 3 : 이 접근법에서는 고차 함수 또는 중첩 함수가 동일한 목적을 달성하는데 사용된다. R에서 함수는 1등 시민으로 여겨지기 때문에 다른 함수의 인수로서 함수를 호출할 수 있다. 폭넓게 사용되는 중첩 함수로는 apply familiy가 있다. 다음 표는 apply familiy에 대한 요약이다.

|  |  |  |
| --- | --- | --- |
| 함수 | 입력 데이터 타입 | 출력 데이터 타입 |
| apply | 데이터 프레임 또는 매트릭스 또는 배열(margin 옵션 필요) | 벡터, 매트릭스, 배열, 리스트 |
| lapply | 벡터, 리스트, 데이터 프레임 또는 매트릭스 내의 변수들 | 리스트 |
| sapply | 벡터, 리스트, 데이터 프레임 또는 매트릭스 내의 변수들 | 매트릭스, 배열, 리스트 |
| mapply  (다변수 sapply) | 벡터, 리스트, 데이터 프레임 또는 매트릭스 내의 변수들 | 매트릭스, 배열, 리스트 |
| tapply | 비정형 배열 | 배열 |
| rapply | 벡터, 리스트, 데이터 프레임 또는 매트릭스 내의 변수들 | 리스트 |

<표 1.4 apply familiy 내의 다양한 함수들>

예제를 통해 이 함수를 확인해보자. apply 함수는 데이터 프레임, 매트릭스 또는 배열에 적용할 수 있다. 매트릭스에 사용해보자.

> x <- cbind(x1 = 7, x2 = c(7:1, 2:5))

> col.sums <- apply(x, 2, sum)

> row.sums <- apply(x, 1, sum)

lapply는 벡터, 리스트 또는 데이터 프레임이나 매트릭스 내의 변수에 적용할 수 있다. lapply 예제는 다음과 같다.

> x <- list(x1 = 7:1, x2 = c(7:1, 2:5))

> lapply(x, mean)

아래는 사용자 정의 함수를 사용하여 벡터에 sapply 함수를 적용한 것이다.

> V\_in <- 1:100000 ## 입력 벡터

> V\_out <- sapply(V\_in, function(x) x^2) ## 출력 벡터

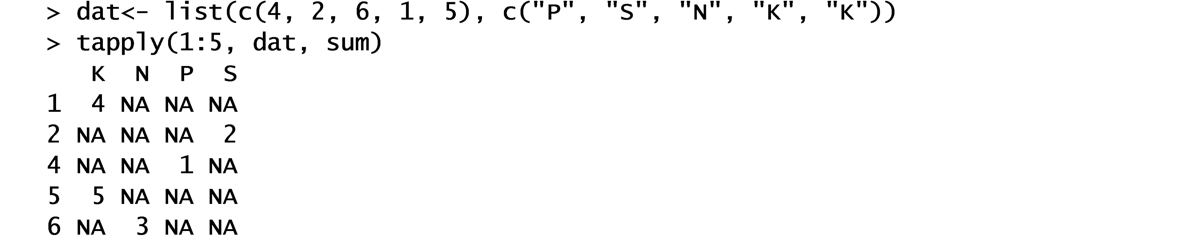
mapply는 다변수 sapply라고 할 수 있다. mapply의 첫번째 입력값은 함수이며, 그 뒤로 다음과 같은 입력 매개변수들이 온다.

mapply(FUN, ..., MoreArgs = NULL, SIMPLIFY = T, USE.NAMES = T)

두 벡터에 대해 rep 함수를 적용한 mapply의 예제는 다음과 같다.

> mapply(rep, 1:6, 6:1)

위 함수는 R의 rep 함수를 호출하여 입력값인 1에서 6까지의 수를 mapply 함수의 두 번째 차원을 이용하여 6번에서 1번으로 줄여가며 복제한다(즉, 1은 6번, 2는 5번, …, 6은 1번 복제). tapply는 비정형 배열의 각 셀에 함수를 적용한다. 다음 예제는 다중 배열로 리스트를 만드는 것이다.



(## 이미지로 된 코드)

위 출력 결과는 각 요소의 위치를 값으로 갖는 두 벡터 간의 관계이다. rapply 함수는 아래 보이는 것처럼 lapply 함수의 재귀함수이다.

> X <- list(list(a = pi, b = list(c = 1:1)), d = "a test")

> rapply(X, sqrt, classes = "numeric", how = "replace")

위 함수는 리스트 내의 모든 숫자형 클래스에 제곱근을 적용하고 그 값을 새로운 값으로 바꾼다.

(숫자형 클래스가 아닌 리스트의 요소에는 영향을 미치지 않는다.)

<대> 연습문제

1. 특정 조건을 기준으로 몇 가지 특성(열)과 관측치(행)를 추출하는 여러 방법을 생각해 낼 수 있는가?

* 데이터셋 - UCI KDD의 엘니뇨 데이터셋(<https://kdd.ics.uci.edu/databases/el_nino/el_nino.html>)
* humidity > 88% 그리고 air temperature < 25.5도인 관측치의 위도와 경도를 추출
* 각 표현식을 검증하기 위해 10,000회 반복

2. apply family의 함수 내에 다양한 인수를 추가할 수 있는가? 그렇다면 여러 개의 인수를 추가하는 문법은 무엇인가?

3. 일반적인 개념에서 for 루프는 apply 함수보다 느리다고 한다. 이것은 사실인가 아닌가? 만약 거짓이라면 이 개념이 무효가 되는 조건은 무엇인가?

4. 원, 사각형 등의 기하학적 객체의 면적을 계산하기 위한 추상 데이터 타입을 정의해보자.

<대> 요약

지난 수십 년 간 컴퓨터의 계산 능력은 지속적으로 증가하고 있으며, 다양한 산업군에서 저장되는 데이터의 양도 역시 증가하고 있다. 데이터의 크기에 대응하기 위해서 빠르고 효율적인 정보 검색이 더욱 요구된다.

1장에서는 추상 데이터 타입과 데이터 구조를 소개했다. 추상 데이터 타입은 다양한 데이터 구조에 대한 기능과 처리를 고수준에서 정의하기 위해 사용되며, 알고리즘은 추상 데이터 타입을 구현하기 위해 사용된다. 데이터 타입은 원자성, 추적 가능성, 정확성을 가져야 하며, 명확성 및 효율성을 위해 분명하고 간결한 특징을 지녀야 한다. 또한 데이터 타입, 조건적 루프, 제어문, 그리고 함수를 포함한 R의 기초를 배웠다.

알고리즘에 따른 계산 시간은 데이터 구조와 알고리즘을 선택할 때 가장 중요한 목표이다. 다음 제2장에서는 알고리즘 분석을 위한 기초적인 내용을 다룰 것이다.