

게임 자료구조 튜터링 교보재

한국산업기술대학교

게임공학과

장 지 웅

게임공학과 전공기초지식 1주			
내용	알고리즘의 개요	선수 연관 과목	프로그래밍
<p>알고리즘이란 어떤 문제를 해결하기 위한 절차 또는 단계들의 유한한 집합을 의미한다. 모든 알고리즘은 다음과 같은 조건들을 만족해야 한다.</p> <p>① 입력: 외부에서 제공되는 자료가 있을 수 있다. 즉, 0개 이상의 입력 자료가 있다.</p> <p>② 출력: 한 가지 이상의 결과를 생성할 수 있어야 한다.</p> <p>③ 명확성: 각 명령어는 명확하고 모호해서는 안 된다.</p> <p>④ 유한성: 한정된 수의 단계 뒤에는 반드시 종료해야 한다.</p> <p>⑤ 유효성: 원칙적으로 모든 명령들은 종이와 연필만으로 수행될 수 있게 기본적인어야 한다. 이것은 각 연산이 명확해서만은 안 되고 반드시 실행 가능해야 한다.</p>			
전공 응용 내용	재귀호출을 이용한 알고리즘 설계	전공 연관 과목	3학년 알고리즘
<p>(예제) $n!$을 구하는 알고리즘을 기술해 보자.</p> <p>(풀이)</p> <pre> int fact1(int n){ 중간 결과값이 저장될 변수를 r 이라고 하자. 반복 회수를 저장할 변수를 counter라고 하자. 단계 1. counter = 0으로 set한다. 단계 2. r = 1으로 set한다. 단계 3. counter = counter + 1 단계 4. r = r * counter 단계 5. if counter < n goto 단계 3. 단계 6. r에 저장된 값을 반환한다. } </pre>			

게임공학과 전공기초지식 1주			
전공 응용 내용	재귀호출을 이용한 알고리즘 설계	전공 연관 과목	3학년 알고리즘
<p>(예제) $n!$을 구하는 재귀적 알고리즘을 기술해 보자.</p> <p>(풀이)</p> <pre> int fact2(int n){ 단계 1. if (n == 0 n == 1) return(1); 단계 2. return(n*fact4(n-1)); } </pre> <p>(예제) 전역변수로 선언된 배열 list[]에 저장된 n개의 값을 곱한 결과를 반환하는 알고리즘 mult1(n)과 동일한 기능을 가진 재귀적 알고리즘 mult2(n)을 기술해 보자.</p> <p>(풀이)</p> <pre> int mult1(int n){ int r; int counter; 단계 1. counter = 0으로 set한다. 단계 2. r = 1으로 set한다. 단계 3. r = r * list[counter] 단계 4. counter = counter + 1 단계 5. if counter < n goto 단계 3. 단계 6. r에 저장된 값을 반환한다. } int mult2(int n){ 단계 1. if (n == 0) return(list[n]); 단계 2. return(mult2(n-1)*list[n]); } </pre>			

게임공학과 전공기초지식 2주			
내용	알고리즘의 성능 분석	선수 연관 과목	프로그래밍
<p>1. 공간복잡도</p> <p>공간복잡도는 크게 고정공간사용량과 가변공간 사용량으로 구성된다. 즉, 다음과 같은 공식이 성립한다.</p> $\text{공간복잡도} = \text{고정공간사용량} + \text{가변공간사용량}$ <p>고정공간사용량이란 프로그램의 입출력 횟수나 크기와 관계없이 프로그램이 수행될 때에 고정적으로 필요한 공간사용량을 의미한다. 이와 같은 고정공간 사용량에는 프로그램 코드를 저장하기 위한 공간, 단순 전역변수, 고정 크기의 구조화 변수, 상수 등이 저장되는 공간이 포함된다.</p> <p>가변공간사용량이란 해결하고자하는 특정 문제(I)에 따르는 크기를 가진 구조화 변수들을 위한 공간들로 구성된다. 예를들어 함수가 재귀호출 되는 경우 재귀호출의 횟수는 프로그램의 입력값에 따라 달라진다고 하면 재귀호출을 처리하기 위해 필요한 공간도 프로그램의 입력값에 따라 변화하게 된다. 이러한 공간사용량을 가변공간사용량이라고 한다.</p> <p>2. 시간복잡도</p> <p>시간복잡도란 프로그램에 의해 소요되는 실행시간을 의미한다. 일반적으로는 프로그램을 구성하는 명령어의 수행 빈도수를 시간복잡도라한다.</p> <p>어떤 프로그램의 시간복잡도를 계산하기 위해서는 단계수 테이블을 사용한다. 단계수 테이블이란 프로그램 실행될 때에 실제로 수행되는 명령어의 빈도 수를 계산하기 위한 표이다.</p>			
전공 응용 내용	시간복잡도 계산	전공 연관 과목	3학년 알고리즘
<p>(예제) 다음 함수의 시간복잡도를 계산하기 위한 단계수 테이블을 작성하라.</p> <pre> float mult1(n) { int fact = 1; int count ; for(count = 0; count < n; count++) fact*=list[count]; return fact; } </pre>			

게임공학과 전공기초지식 2주

전공 응용 내용

시간복잡도 계산

전공 연관 과목

3학년 알고리즘

(폴이)

문장	s/e	빈도수	총단계수
float mult1(n) {	0	0	0
int fact = 1;	1	1	1
int count ;	0	0	0
for(count = 0; count < n; count++)	1	n+1	n+1
fact*=list[count];	1	n	n
return fact;	1	1	1
}	0	0	0
합계			2n+3

(예제) 동일한 기능을 가진 다음 함수의 단계수 테이블을 작성하라.

```
float mult(n) {
    if(n == 0)
        return(list[0]);
    return(mult(n-1)*list[n-1]);
}
```

(폴이)



문장	s/e	빈도수	총단계수
float mult(n) {	0	0	0
if(n == 0)	1	n+1	n+1
return(list[0]);	1	1	1
return(mult(n-1)*list[n-1]);	1	n	n
}	0	0	0
합계			2n+2

게임공학과 전공기초지식 3주			
내용	배열을 이용한 데이터 관리	선수 연관 과목	프로그래밍
<p>배열이란 비슷한 구조의 데이터들을 하나의 변수 이름을 사용해서 만든 여러 개의 논리적이며 물리적인 연속된 기억장소를 의미한다. 배열을 사용하면 동일한 타입의 변수 여러개를 일일이 선언하지 않고 한꺼번에 선언할 수 있으며, 반복문을 사용하여 동일한 방식의 연산을 수행하여 일괄적으로 처리할 수 있다는 장점이 있다.</p> <p>배열의 특징은 다음과 같다.</p> <ol style="list-style-type: none"> 1. 원소들간에 논리적인 순서가 있고, 물리적인 위치도 논리적인 순서와 동일한 순서로 정해진다. 2. 자료를 저장, 추출하는 것은 자료의 위치와 관계없이 동일한 시간에 처리가 가능하지만, 삽입과 삭제의 경우에는 논리적인 순서를 지키기 위하여 위치 이동이 필요하므로 삽입, 삭제 위치에 따라 수행시간이 다르다. 			
전공 응용 내용	배열을 이용한 데이터 관리	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열에 저장된 모든 Data를 출력하는 AllPrint 기능을 구현하시오.</p> <p>(풀이)</p> <pre> struct Node{ int hp; int attack; int defence; char name[20]; }; void AllPrint(Node *pNodeList) { for(int i = 0; i < CurrentIndex; i++) { printf(" %d, %d, %d, %s \n", pNodeList[i].hp, pNodeList[i].attack, pNodeList[i].defence, pNodeList[i].name); } } </pre>			

게임공학과 전공기초지식 3주			
전공 응용 내용	배열을 이용한 데이터 관리	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열에 저장된 Data에서 원하는 Data를 검색하는 Search 기능을 구현하시오. (폴이)</p> <pre> bool Search(Node *pNodeList) { char name[20]; scanf("%s", name); for(int i = 0; i < CurrentIndex; i++) { if(!strcmp(pNodeList[i].name, name)) { printf(" %d, %d, %d, %s Wn", pNodeList[i].hp, pNodeList[i].attack, pNodeList[i].defence, pNodeList[i].name); return true; } } return false; } </pre> <p>(예제) 배열에 새로운 Data를 추가하는 Insert 기능을 구현하시오. (폴이)</p> <pre> bool Insert(Node* pNodeList) { int hp, attack, deffence; char name[20]; scanf("%d %d %d %s", &hp, &attack, &deffence, name); return Insert(hp, attack, deffence, name, pNodeList); } bool Insert(int hp, int attack, int defence, char* name, Node *pNodeList) { if(CurrentIndex < 50) { pNodeList[CurrentIndex].hp = hp; pNodeList[CurrentIndex].attack = attack; pNodeList[CurrentIndex].defence = defence; strcpy(pNodeList[CurrentIndex].name, name); ++ CurrentIndex; return true; } return false; } </pre>			

게임공학과 전공기초지식 4주			
내용	배열을 이용한 데이터 관리	선수 연관 과목	프로그래밍
<p>배열에 저장된 특정 데이터를 삭제하려고 하는 연산의 다음의 단계에 따라 수행된다.</p> <ol style="list-style-type: none"> 1. 삭제하려는 데이터가 저장된 배열상에서의 위치를 탐색한다. 2. 데이터를 삭제한다. 3. 배열에서 삭제된 데이터보다 뒤쪽에 저장된 데이터를 한 칸씩 앞으로 이동시킨다. <p>이때 단계 3의 연산은 매우 비용이 비싼 연산에 해당한다. 왜냐하면 데이터의 개수가 N개 일 때 최악의 경우 N-1개의 데이터를 위치 이동하여야 하기 때문이다.</p>			
전공 응용 내용	배열에 저장된 데이터의 삭제	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열에 저장된 Data 중에서 원하는 Data를 삭제하는 Delete 기능을 구현하시오. (풀이)</p> <pre> bool Delete(Node* pNodeList) { char name[20]; scanf("%s", name); for(int i = 0; i < CurrentIndex; i++) { if(!strcmp(pNodeList[i].name, name)) { for(int j = i; j <= CurrentIndex; j++) memcpy(&pNodeList[j], &pNodeList[j + 1], sizeof(Node)); --CurrentIndex; return true; } } return false; } </pre>			

게임공학과 전공기초지식 4주			
전공 응용 내용	배열에 저장된 데이터의 삭제	전공 연관 과목	3학년 알고리즘
<p>(예제) 다음에 소개하는 구조체 UnitStruct는 어떤 게임의 캐릭터에 대한 정보를 저장하는 자료구조이다. UnitStruct형 배열 unit[]에 n개의 데이터가 저장되어 있을 때 이 중에서 hp가 가장 큰 unit의 speed를 가진 유닛을 삭제하는 함수 del_strongest(UnitStruct unit[], n)을 작성하라.</p> <pre> typedef struct { int hp; // 체력 int mp; // 마법력 int speed; // 속도 int attack_point; // 공격력 int defence_point // 방어력 } UnitStruct; </pre> <p>(풀이)</p> <pre> void del_strongest(UnitStruct unit[], int n) { int i; int j; int top = 0; // hp가 가장 높은 데이터를 찾아 그 인덱스를 top에 저장한다. for(i = 1; i < n; i++) if(unit[i].hp > unit[top].hp) top = i; // top의 위치에 저장된 데이터를 삭제한다. for(int j = top; j < n; j++) memcpy(&pNodeList[j], &pNodeList[j + 1], sizeof(UnitStruct)); return; } </pre>			

게임공학과 전공기초지식 5주			
내용	배열에서의 문자열 관리	선수 연관 과목	프로그래밍
<p>프로그램에서 배열을 사용하기 위해서는 먼저 배열을 선언하여야 한다. 이 때 배열을 구성하는 원소의 데이터 타입과 배열의 크기, 이름을 명시하여야 한다. 다음은 10개의 정수형 데이터 원소를 가지는 배열 arr과 5개의 문자형 데이터 원소를 가지는 배열 str의 선언 예이다.</p> <pre>int arr[10]; char str[5];</pre> <p>이와 같이 선언되는 경우 메모리 상에서는 arr과 str을 위한 저장공간이 설정된다. 이 때 하나의 배열에 속하는 원소들은 물리적으로도 인접한 위치에 저장공간이 설정된다. 다음은 배열 str의 물리적인 저장 위치를 도식화한 것이다. 그림에서 str의 원소 5개는 물리적으로도 인접한 위치에 설정되었음을 알 수 있다.</p> <div style="text-align: center;"> <pre>str[0] str[1] str[2] str[3] str[4]</pre>  </div> <p>특별히, 문자열은 문자형의 배열을 이용하여 표현한다. 이 때, 문자열의 마지막 원소에는 반드시 '\0'이 저장된다. 다음은 문자열에 대한 선언 및 초기화의 예이다.</p> <pre>char string[16] = "Korea";</pre> <p>다음은 위와 같이 초기화한 문자열의 실제 저장 상태를 도식화 한 것이다.</p> <div style="text-align: center;"> <pre>string[0] string[1] string[2] string[3] string[4] string[5]</pre>  </div>			
전공 응용 내용	배열을 이용한 문자열 처리 방법	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열 arr에 저장된 값을 순서대로 출력하는 프로그램을 작성하라.</p> <p>(풀이)</p> <pre>#include <stdio.h> main() { int arr[5] = "1,2,3,4,5"; int i; for(i = 0; i < 5; i++) printf("%d ", arr[i]); }</pre>			

게임공학과 전공기초지식 5주			
전공 응용 내용	배열을 이용한 문자열 처리 방법	전공 연관 과목	3학년 알고리즘
<p>(예제) 10자 이내의 두개의 문자열을 입력받아 하나로 연결하여 출력하는 프로그램을 작성하라.</p> <p>(풀이)</p> <pre> #include <stdio.h> main() { char string1[11]; // 첫 번째 문자열 char string2[11]; // 두 번째 문자열 char string[22]; // 결과 문자열 int i, j; // 첫 번째 문자열을 입력받는다. scanf("%s", string1); // 두 번째 문자열을 입력받는다. scanf("%s", string2); // 첫 번째 문자열을 결과 문자열에 복사한다. for(i=0; string1[i] != '\0'; i++) string[i] = string1[i]; // 두 번째 문자열을 결과 문자열에 복사한다. for(j=0; string2[j] != '\0'; i++, j++) string[i] = string2[j]; // 결과 문자열을 출력한다. printf("%s", string); } </pre>			

게임공학과 전공기초지식 6주			
내용	배열을 이용한 문자열 관리의 응용	선수 연관 과목	프로그래밍
<p>배열은 가장 기본적인 자료구조로서 배열의 활용력을 높이는 것은 자료구조를 공부하는 초석이 된다. 그러므로 본 실습에서는 배열을 이용하여 문자열을 변형하고, 관리하는 방법에 대하여 실습문제를 해결하는 방식으로 연습한다.</p>			
전공 응용 내용	배열을 이용한 문자열 관리 실습	전공 연관 과목	3학년 알고리즘
<p>(예제) 하나의 문자열을 입력받아서 그것을 거꾸로 만드는 함수 reverse(char a[], char b[], size)을 작성하시오.</p> <p>(풀이)</p> <pre>void reverse(char a[], char b[], int size) { int i, j; for(i = 0; a[i] != '\0'; i++); for(j=0; i > 0; i--, j++) b[j] = a[i-1]; b[j] = '\0'; }</pre> <p>(예제) 위에서 작성한 함수 reverse()와 동일한 작업을 수행하는 reverse2()를 재귀호출을 이용하여 작성하시오.</p> <p>(풀이)</p> <pre>void reverse2(char a[], char b[], int size) { if(size == 0) return; else { b[0] = a[size-1]; reverse2(a, b+1, size-1); } }</pre>			

게임공학과 전공기초지식 6주			
전공 응용 내용	배열을 이용한 문자열 관리 실습	전공 연관 과목	3학년 알고리즘
<p>(예제) 다음과 같이 출력되는 프로그램을 작성하라.</p> <pre> 1 16 15 14 13 2 17 24 23 12 3 18 25 22 11 4 19 20 21 10 5 6 7 8 9 </pre> <p>(풀이)</p> <pre> #include<stdio.h> int k, h = 0; int y = -1; int suja = 0; int byul[20][20]; void cycle(int n, int d){ if(n == 0) return; for (k=0; k < 2*n-1; k++){ if(k < n) y+=d; else h+=d; suja ++; byul[h][y] = suja; } cycle(n-1, d*(-1)); } main(){ int p,j,k; cycle(5, 1); for(p = 0; p < 5; p++){ for(j=0; j < n; j++) printf("%3d ", byul[p][j]); printf("\n"); } } </pre>			

게임공학과 전공기초지식 7주			
내용	재귀함수를 활용한 데이터 관리	선수 연관 과목	프로그래밍
<p>재귀함수를 이용하면 배열에 저장된 데이터의 관리를 좀 더 간편하게 처리할 수 있다. 그러나 재귀함수를 능숙하게 다루기 위해서는 재귀적인 사고 방식이 필요하므로 여러 예제 프로그램을 작성함으로써 재귀적 사고력을 높이는 것이 무엇보다 중요하다. 본 실습에서는 재귀함수를 이용하여 배열에 저장된 데이터를 처리하는 방법을 실습함으로써 재귀함수의 활용력을 높인다.</p>			
전공 응용 내용	재귀함수를 이용한 배열 관리	전공 연관 과목	3학년 알고리즘
<p>(예제) 재귀함수를 이용하여 체력이 가장 큰 유닛을 찾는 함수를 구현하시오.</p> <p>(폴이)</p> <pre> struct Node{ int hp; int attack; int defence; char name[20]; }; void swap(Node &lhs, Node &rhs) { Node temp; memcpy(&temp, &lhs, sizeof(Node)); memcpy(&lhs, &rhs, sizeof(Node)); memcpy(&rhs, &temp, sizeof(Node)); } </pre>			

게임공학과 전공기초지식 7주			
전공 응용 내용	재귀함수를 이용한 배열 관리	전공 연관 과목	3학년 알고리즘
<pre> int ResursiveMaxData(Node *pNodeList, int rhs) { if(rhs <= 0) { if(pNodeList[rhs].hp > pNodeList[rhs + 1].hp) return rhs; else return rhs + 1; } int lhs = ResursiveMaxData(pNodeList , rhs - 1); if(pNodeList[lhs].hp > pNodeList[rhs].hp) return lhs; else return rhs; } </pre> <p>(예제) 재귀함수만을 사용하여 배열에 있는 모든 Data를 정렬하는 함수를 구현하라.</p> <p>(풀이)</p> <pre> void ResursiveSort(Node *pNodeList, int num) { if(num <= 0) return; int maxIndex = 0; maxIndex = ResursiveMaxData(pNodeList, num); swap(pNodeList[maxIndex], pNodeList[num]); ResursiveSort(pNodeList, num - 1); } </pre>			

게임공학과 전공기초지식 8주			
내용	배열을 이용한 게임의 구현(1)	선수연관과목	프로그래밍
배열에 화면 정보를 저장하고, 키보드 입력을 이용하여 저장된 정보를 조작, 출력함으로써 텍스트 화면에서의 게임을 구현할 수 있다. 이러한 방식으로 구현된 게임은 향후 그래픽 화면 처리를 이용하여 2D 게임이나 3D 게임으로의 전환이 가능하다.			
전공 응용 내용	배열을 이용한 텍스트 게임의 구현(1) - 화면 구성	전공연관과목	3학년 알고리즘
<p>(예제) 배열을 이용하여 텍스트 화면에서 전통적인 유명 게임인 팩맨을 구현해 보자. 팩맨의 조정은 키보드 입력을 이용한다. 본 실습에서는 게임 화면을 구성하라.</p> <p>(폴이)</p> <pre> #include <stdio.h> #include <windows.h> #include <conio.h> #include <time.h> #include <iostream> using namespace std; int x=3,y=5,x2=6,knownx=0,knowny=0; int monx[4]={1,18,1,18},mony[4]={1,1,18,18}; #define LEFT 'a' #define RIGHT 'd' #define DOWN 's' #define UP 'w' void mon_move(); HANDLE g_hOutput; int jumpsu; bool CanKill = false; int xy[20][20] = { 0, 0,1,1,1,1,1,1,1,1,0,0,1,2,1,1,1,1,1,1,0, 0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,0,1,0, 0,1,0,0,0,1,0,0,1,1,1,2,0,0,1,0,0,0,0,1,0, 0,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,0, 0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,0,1,0, 0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0, 0,1,0,0,0,1,0,0,0,0,0,9,0,0,1,0,0,0,0,1,0, 0,1,0,0,0,1,0,9,9,9,9,9,0,1,0,0,0,1,0, 1,1,1,1,1,1,0,9,9,2,2,9,9,0,1,1,1,1,1,1, 0,1,0,0,0,1,0,9,9,9,9,9,0,1,0,0,0,1,0, 0,1,0,0,0,1,0,9,9,9,9,9,0,1,0,0,0,1,0, 0,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,0, 0,1,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,1,0, 0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,1,0, 0,1,1,1,1,1,0,0,1,0,0,1,0,0,1,1,1,1,1,0, 0,1,0,0,0,1,1,1,2,2,1,1,1,1,1,0,0,0,1,0, 0,1,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,0,1,0, 0,1,1,1,1,1,2,1,1,0,0,1,1,1,1,1,1,1,0, 0,}; </pre>			

게임공학과 전공기초지식 8주			
전공 응용 내용	배열을 이용한 텍스트 게임의 구현(1) - 화면 구성	전공연관과목	3학년 알고리즘
<pre> void sleep(double delaytime){ clock_t delay; delay=(delaytime * CLOCKS_PER_SEC) +clock(); while(delay>clock()){ /*delay가 현재시간 보다 크면 루프를 돌려서 시간을 지연 시킨다. */ } } void gotoxy(int x,int y); void screen(){ int i,j; system("cls"); for(i=0;i<20;i++){ for(j=0;j<20;j++){ if(xy[i][j] == 0){ SetConsoleTextAttribute(g_hOutput, 134); printf(" "); } else if (xy[i][j] == 1){ SetConsoleTextAttribute(g_hOutput, 127); printf(" . "); } else if (xy[i][j] == 2){ SetConsoleTextAttribute(g_hOutput, 127); printf("⊙"); } else if (xy[i][j] == 3){ SetConsoleTextAttribute(g_hOutput, 124); printf("▼");//(추가)몬스터 } else if (xy[i][j] == 4){ SetConsoleTextAttribute(g_hOutput, 125); printf("▼");//(추가)몬스터 } else if (xy[i][j] == 5){ SetConsoleTextAttribute(g_hOutput, 123); printf("▼");//(추가)몬스터 } else if (xy[i][j] == 6){ SetConsoleTextAttribute(g_hOutput, 122); printf("▼");//(추가)몬스터 } else if (xy[i][j] == 9){ SetConsoleTextAttribute(g_hOutput, 119); printf(" ");//빈 벽 } } printf("\n"); } SetConsoleTextAttribute(g_hOutput, 112); printf("점수 : %d", jumsu); SetConsoleTextAttribute(g_hOutput, 126); gotoxy(x2,y); printf("●"); } </pre>			


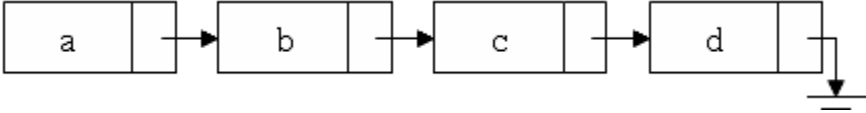
게임공학과 전공기초지식 9주			
내용	배열을 이용한 게임의 구현(2)	선수 연관 과목	프로그래밍
<p>배열에 화면 정보를 저장하고, 키보드 입력을 이용하여 저장된 정보를 조작, 출력함으로써 텍스트 화면에서의 게임을 구현할 수 있다. 이러한 방식으로 구현된 게임은 향후 그래픽 화면 처리를 이용하여 2D 게임이나 3D 게임으로의 전환이 가능하다.</p>			
전공 응용 내용	배열을 이용한 텍스트 게임의 구현(2)-키보드입력	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열을 이용하여 텍스트 화면에서 전통적인 유명 게임인 팩맨의 키보드 입력 부분을 구현하라.</p> <p>(폴이)</p> <pre> void input() { char c; int z=1; while(z){ c=getch(); while(z){ int cnt = rand()%3; switch(c){ case UP: --y; break; case RIGHT: ++x; x2 += 2; break; case LEFT: --x; x2 -=2; break; case DOWN: ++y; break; } if(xy[y][x] == 1){ xy[y][x] = 9; jumsu += 10; } if(xy[y][x] == 2){ xy[y][x] = 9; jumsu += 50; CanKill = true; } if(CanKill = true && xy[y][x] ==3){ xy[y][x] = 9; monx[0] = mony[0] = -1; CanKill = false; } if(CanKill = true && xy[y][x] ==4) { xy[y][x] = 9; monx[1] = mony[1] = -1; CanKill = false; } if(CanKill = true && xy[y][x] ==5){ xy[y][x] = 9; monx[2] = mony[2] = -1; CanKill = false; } } } </pre>			

게임공학과 전공기초지식 9주			
전공 응용 내용	배열을 이용한 텍스트 게임의 구현(2)-키보드입력	전공 연관 과목	3학년 알고리즘
<pre> if(CanKill = true && xy[y][x] ==6){ xy[y][x] = 9; monx[3] = mony[3] = -1; CanKill = false; } sleep(0.6); switch(c){ case UP: if(xy[y][x] == 0) ++y; break; case RIGHT: if(xy[y][x] == 0){ --x; x2 -=2; } if(x2 == 38){ x2 = 0; x=0; y=9; } break; case LEFT: if(xy[y][x] == 0) { ++x; x2+=2; } if(x2 == 0){ x2 = 38; x=19; y=9; } break; case DOWN: if(xy[y][x] == 0) --y; break; } if(cnt==0) mon_move(); screen(); gotoxy(x2,y); if(kbhit()) break; } //end of while z=0; } // end of while } </pre>			

게임공학과 전공기초지식 10주			
내용	배열을 이용한 게임의 구현(3)	선수 연관 과목	프로그래밍
<p>배열에 화면 정보를 저장하고, 키보드 입력을 이용하여 저장된 정보를 조작, 출력함으로써 텍스트 화면에서의 게임을 구현할 수 있다. 이러한 방식으로 구현된 게임은 향후 그래픽 화면 처리를 이용하여 2D 게임이나 3D 게임으로의 전환이 가능하다.</p>			
전공 응용 내용	배열을 이용한 텍스트 게임의 구현(3)-괴물이동	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열을 이용하여 텍스트 화면에서 전통적인 유명 게임인 팩맨의 괴물이동 부분을 구현하라.</p> <p>(풀이)</p> <pre> void mon_move(){ for(int i=3; i<3+4; i++) { xy[mony[i-3]][monx[i-3]]=i; int a=y-mony[i-3]; int b=x-monx[i-3]; if(a>0){ if(xy[mony[i-3]+1][monx[i-3]]!=0){ xy[mony[i-3]][monx[i-3]]=1; mony[i-3]+=1; if(mony[i-3]==y){ if(monx[i-3]==x) exit(1); } xy[mony[i-3]][monx[i-3]]=i; } } if(a<0){ if(xy[mony[i-3]-1][monx[i-3]]!=0){ xy[mony[i-3]][monx[i-3]]=1; mony[i-3]-=1; if(mony[i-3]==y){ if(monx[i-3]==x) exit(1); } xy[mony[i-3]][monx[i-3]]=i; } } if(b>0){ if(xy[mony[i-3]][monx[i-3]+1]!=0){ xy[mony[i-3]][monx[i-3]]=1; monx[i-3]+=1; if(mony[i-3]==y){ if(monx[i-3]==x) exit(1); } xy[mony[i-3]][monx[i-3]]=i; } } } } </pre>			

게임공학과 전공기초지식 10주			
전공 응용 내용	배열을 이용한 텍스트 게임의 구현(3)-괴물이동	전공 연관 과목	3학년 알고리즘
<pre> if(b<0){ if(xy[mony[i-3]][monx[i-3]-1]!=0){ xy[mony[i-3]][monx[i-3]]=1; monx[i-3]-=1; if(mony[i-3]==y){ if(monx[i-3]==x) exit(1); } xy[mony[i-3]][monx[i-3]]=i; } } } } void getwall(int a,int b,int c){ int d; for(;b<c;b++){ xy[a][b]=1; srand((unsigned)time(NULL)); d=rand()%c; if(b<d){xy[a][d-1]=2;} } } void getwall2(int a,int b,int c){ int d; for(;b<c;b++){ xy[b][a]=1; srand((unsigned)time(NULL)); d=rand()%c; if(a<d){xy[b][d]=2;} } } void main(){ g_hOutput = GetStdHandle(STD_OUTPUT_HANDLE); srand((unsigned int)time(NULL)); while(1){ screen(); input(); for(int i=0; i<4; i++) if(monx[i] == -1) monx[i] = mony[i] = 9; } } </pre>			

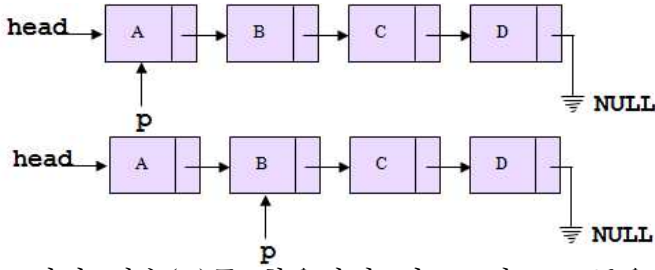
게임공학과 전공기초지식 11주

내용	리스트	선수 연관 과목	프로그래밍
<p>리스트는 데이터 원소(노드)의 논리적인 순서만 유지될 뿐 데이터의 실제 물리적 저장 위치는 논리적인 순서와 무관하게 결정되는 자료구조이다. 그러므로 데이터 원소의 논리적인 순서를 유지하기 위한 별도의 자료구조가 필요하다. 일반적으로 리스트에 속하는 데이터 원소는 데이터의 값과 함께 논리적인 순서에 따라 다음 데이터가 저장되어 있는 메모리 상 위치의 주소값을 가지고 있다.</p> <p>다음 그림은 리스트의 노드에 대한 자료구조와 그 형태를 도식화 한 것이다. 그림에서 data는 노드에 저장된 데이터의 값이고, next는 논리적인 순서 상 다음 노드가 저장된 물리적 위치를 나타낸다.</p> <p>동일한 방식으로 여러 노드를 연결하면 논리적인 순서가 유지되는 리스트를 만들 수 있다. 다음 그림은 여러 개의 노드가 연결되어 단방향 리스트가 구성된 형태를 도식화 한 것이다.</p> <div style="text-align: center;"> <pre>typedef struct{ char data; Node *next; } Node</pre> <div style="display: inline-block; vertical-align: middle; text-align: center;"> <p>Node</p>  <p>next</p> </div> </div> <p style="text-align: center;">리스트를 구성하는 노드의 구조.</p> <div style="text-align: center; margin-top: 20px;">  </div> <p style="text-align: center;">단방향 리스트의 형태.</p>			

게임공학과 전공기초지식 11주			
전공 응용 내용	단방향 리스트	전공 연관 과목	3학년 알고리즘
<p>(예제) 단방향 리스트의 노드를 초기화 하는 함수 Node *init_list(data)를 작성하고, init_list()를 활용하여 'K', 'O', 'R', 'E', 'A'가 순서대로 저장된 단방향 리스트를 하나 생성하는 프로그램을 작성하라. 이 때 init_list()는 노드를 생성하여 주어진 data의 값을 저장하는 작업을 수행한다.</p> <p>(풀이)</p> <pre> Node *init_node(char data) { Node *p; // 새로운 node를 생성하기 위해 메모리를 할당받는다. p = (Node *)malloc(sizeof(Node)); // 생성된 node의 data 부분에 주어진 data를 저장한다. p->data = data; // 생성된 node의 next는 아직 가리키는 것이 없으므로 NULL 로 설정한다. p->next = NULL; // 생성된 node를 가리키는 포인터 변수 p를 반환한다. return(p); } int main() { Node *head; // 리스트의 첫 번째 노드를 가리킬 포인터 // head에 새로운 노드를 생성하여 연결한다. head = init_node('K'); head->next = init_node('O'); head->next->next = init_node('R'); head->next->next->next = init_node('E'); head->next->next->next->next = init_node('A'); } </pre>			

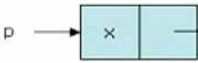
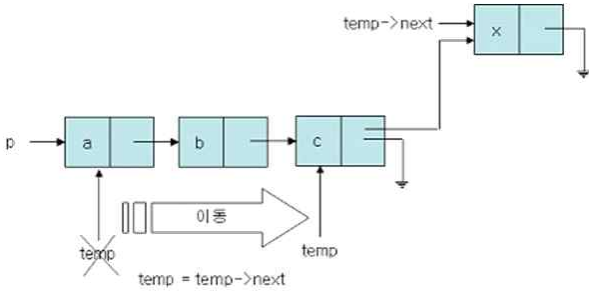
게임공학과 전공기초지식 12주			
내용	포인터의 이해	선수 연관 과목	프로그래밍
<p>포인터형 변수란 메모리상의 물리적 주소를 값으로 가지는 변수를 의미한다. 포인터를 이용하면 배열이 가지는 물리적으로 인접해야한다는 제약 조건을 탈피할 수 있으므로 매우 유용하게 사용된다.</p> <p>포인터형 변수란 메모리상의 물리적 주소를 값으로 가지는 변수이다. 포인터형 변수에 대한 기본 연산자로는 *(pointer-following)연산자와 &(address-of)연산자가 있다.</p> <p>다음의 예제 프로그램을 작성하여 수행해 보자.</p> <pre>main(){ int n; int *p; p = &n; // 변수 p에 변수 n의 주소값을 저장한다. *p = 10; // 변수 p에 저장된 주소에 10을 저장한다. printf("n = %d", n); }</pre> <p>위의 예제 프로그램에서 p는 포인터형 변수이다. “p = &n”은 변수 p에 변수 n의 주소값을 저장하라는 의미이며, “*p = 10”은 변수 p에 저장된 주소에 10을 저장하라는 의미이다. 현재 변수 p에는 변수 n의 주소값이 저장되어 있으므로 결과적으로 변수 n에 10을 저장하라는 것과 동일한 의미가 된다. 즉, 본 프로그램을 수행하면 “n = 10”이 출력된다.</p>			
전공 응용 내용	포인터를 이용한 배열 데이터 관리	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열을 포인터 변수를 이용하여 원하는 Data를 검색할 수 있는 Search 기능을 구현하시오.</p> <p>(풀이)</p> <pre>bool Search(Node *pNodeList){ char name[20]; scanf("%s", name); for(int i = 0; i < CurrentIndex; i++) { if(!strcmp(pNodeList[i].name, name)) {</pre>			

게임공학과 전공기초지식 12주			
전공 응용 내용	포인터를 이용한 배열 데이터 관리	전공 연관 과목	3학년 알고리즘
<p>(폴이)</p> <pre> printf(" %d, %d, %d, %s \n", pNodeList[i].hp, pNodeList[i].attack, pNodeList[i].defence, pNodeList[i].name); return true; } } return false; } </pre> <p>(예제) 배열을 포인터 변수를 이용하여 모든 Data를 정렬하는 기능을 구현하라.</p> <p>(폴이)</p> <pre> void swap(Node &lhs, Node &rhs){ Node temp; memcpy(&temp, &lhs, sizeof(Node)); memcpy(&lhs, &rhs, sizeof(Node)); memcpy(&rhs, &temp, sizeof(Node)); } bool Sort(Node *pNodeList){ int nMaxIndex; for(int i = 0; i < CurrentIndex; i++) { nMaxIndex = i; for(int j = 0; j < CurrentIndex - i; j++) { if(pNodeList[j] > pNodeList[nMaxIndex]) nMaxIndex = j; } swap(pNodeList[nMaxIndex], pNodeList[CurrentIndex - 1]); } return false; } </pre>			

게임공학과 전공기초지식 13주			
내용	단방향 리스트의 검색	선수 연관 과목	프로그래밍
<p>단방향 연결리스트에서 원하는 데이터를 검색하는 경우 포인터가 단일 방향으로만 연결되어 있으므로 해당 방향으로의 검색만이 가능하다. 다음 그림은 단방향 연결리스트에서의 검색 과정을 도식화 한 것이다.</p>  <p>그림과 같이 별도의 포인터 변수(p)를 활용하여 리스트의 노드들을 하나씩 탐색하는 방법으로 원하는 데이터를 검색할 수 있다.</p>			
전공 응용 내용	단방향 연결리스트에서의 검색	전공 연관 과목	3학년 알고리즘
<p>(예제) 단방향 연결 리스트에서 원하는 Data를 검색하는 Search 기능을 구현하시오.</p> <p>(풀이)</p> <pre> struct node { char c; long cnt; int hp; int attack; int def; int range; int mana; struct node *next; }; struct node *head; void Search(void) { </pre>			

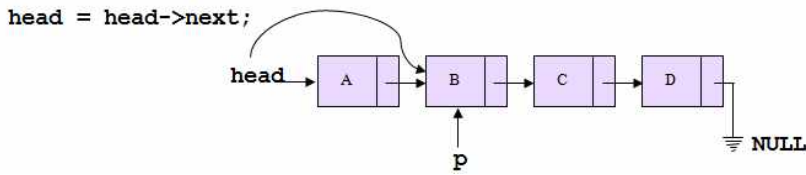
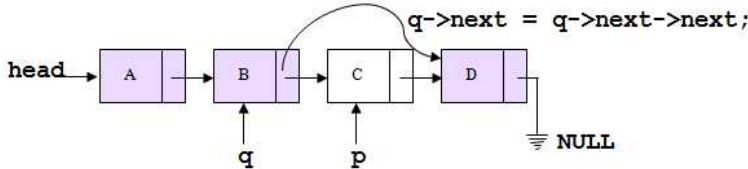
게임공학과 전공기초지식 13주			
전공 응용 내용	단방향 연결리스트에서의 검색	전공 연관 과목	3학년 알고리즘
<pre> struct node *temp; long i=0; temp=head->next; printf("찾으시는 번호를 입력하세요 : "); scanf("%ld%c",&i, &dummy); while(temp!=NULL) { if(temp->cnt==i) { print_unit(temp); break; } temp=temp->next; } system("pause"); } </pre> <p>(예제) 배열 Search 수행 시간과 리스트 Search 수행 시간을 표시하고 비교하시오.</p> <p>(폴이)</p> <pre> void searcharr(void) { int sel; int i=0; long cnt=0; clock_t start, end; struct UNIT temp; printf("찾는 체력을 입력: "); scanf("%d%c", &sel, &dummy); start=clock(); while(cnt<MAX) { if(U[cnt].hp == sel) { temp=U[cnt]; break; } cnt++; } while(cnt<MAX) { </pre>			

게임공학과 전공기초지식 13주			
전공 응용 내용	단방향 연결리스트에서의 검색	전공 연관 과목	3학년 알고리즘
<pre> if(U[cnt].hp == temp.hp) { print_arrunit(U[cnt]); i+ +; } cnt+ +; } if(i==0) printf("찾는 유닛이 없습니다.\n"); end=clock(); printf("%.3lf초!\n", (double)(end-start) /CLOCKS_PER_SEC); system("pause"); } void searchlinked(void) { clock_t start, end; struct node *temp; long sel; int i=0; printf("찾을 체력을 입력하세요: "); scanf("%ld%c",&sel, &dummy); start=clock(); temp=head; while(temp!=NULL) { if(sel==temp->hp) { print_unit(temp); i+ +; } temp=temp->next; } if(i==0) printf("찾는 유닛이 없습니다.\n"); end=clock(); printf("%.3lf초!\n", (double)(end-start) / CLOCKS_PER_SEC); system("pause"); } </pre>			

게임공학과 전공기초지식 14주			
내용	연결리스트에서의 삽입	선수 연관 과목	프로그래밍
<p>연결리스트에 새로운 노드를 삽입하는 연산은 크게 리스트가 비어 있는 경우의 삽입과 그렇지 않은 경우의 삽입으로 구분된다. 다음 그림은 두가지 경우의 삽입 방법을 도식화 한 것이다. 본 예에서는 리스트가 비어있지 않은 경우 세 번째 노드의 뒤에 새로운 노드를 삽입하는 것을 가정하였다.</p> <div data-bbox="611 710 1021 931"> <p>$p \rightarrow \text{NULL}$</p> <p>'x' 라는 데이터값을 가지는 노드를 생성</p>  </div> <p>리스트가 비어있는 경우</p> <div data-bbox="509 1075 1101 1366">  </div> <p>리스트가 비어있지 않은 경우</p>			
전공 응용 내용	연결리스트에서 데이터의 삽입	전공 연관 과목	3학년 알고리즘
<p>(예제) 연결 리스트에서 새로운 Data를 추가하는 Insert 기능을 구현하시오. (풀이)</p>			

게임공학과 전공기초지식 14주			
전공 응용 내용	연결리스트에서 데이터의 삽입	전공 연관 과목	3학년 알고리즘
<pre> (폴이) void addnode(void) { struct node *New, *temp; long i=0; temp=head->next; New=(struct node *)malloc(sizeof(struct node)); while(temp!=NULL) { if(i<temp->cnt) i=temp->cnt; temp=temp->next; } printf("종족 : "); scanf("%c%c", &New->c, &dummy); printf("체력 : "); scanf("%d%c", &New->hp, &dummy); printf("공격력 : "); scanf("%d%c", &New->attack, &dummy); printf("방어력 : "); scanf("%d%c", &New->def, &dummy); printf("시야 : "); scanf("%d%c", &New->range, &dummy); printf("마나 : "); scanf("%d%c", &New->mana, &dummy); New->cnt=i+ 1; printf("완료되었습니다!\n"); temp=head; while(temp->next!=NULL) { if(New->hp>temp->next->hp) break; else temp=temp->next; } New->next=temp->next; temp->next=New; system("pause"); } </pre>			

게임공학과 전공기초지식 15주

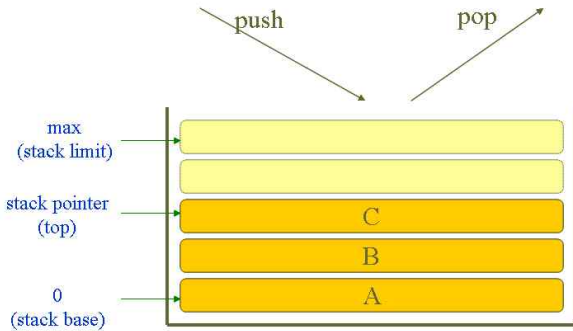
내용	연결리스트에서의 삭제	선수 연관 과목	프로그래밍
<p>연결리스트에서 특정 노드를 삭제하는 연산은 크게 리스트가 리스트의 첫 노드를 삭제하는 경우와中间的의 노드(또는 마지막 노드)를 삭제하는 경우로 구분된다. 다음 그림은 두가지 경우의 삭제 방법을 도식화 한 것이다.</p> <p>첫 노드를 삭제하는 경우에는 head 포인터를 그 다음 노드로 연결하는 것으로 삭제가 완료되고, 중간 노드를 삭제하는 경우에는 삭제하려는 노드의 앞 노드에 있는 포인터를 다음 노드로 연결하여야 한다.</p>			
<div style="text-align: center;">  <p>첫 노드를 삭제하는 경우</p> </div>			
<div style="text-align: center;">  <p>중간 노드를 삭제하는 경우</p> </div>			
전공 응용 내용	연결리스트에서 데이터의 삭제	전공 연관 과목	3학년 알고리즘
<p>(예제) 연결 리스트에서 원하는 Data를 삭제하는 Delete 기능을 구현하시오.</p> <p>(폴이)</p> <pre> void Deletenode(void) { struct node *del; struct node *temp; long i=0; temp = head; printf("찾으시는 번호를 선택하세요 : "); scanf("%ld%c", &i, &dummy); while(temp!=NULL) { del=temp->next; if(del->cnt==i) { temp->next=del->next; i=-1; free(del); break; } } } </pre>			

게임공학과 전공기초지식 15주			
전공 응용 내용	연결리스트에서 데이터의 삭제	전공 연관 과목	3학년 알고리즘
<p>(폴이)</p> <pre> temp->next=del->next; i=-1; free(del); break; } temp=temp->next; } if(i>=0) printf("찾으시는 번호가 없습니다\n"); else printf("완료되었습니다\n"); system("pause"); } </pre> <p>(예제) 환형 리스트에서 노드의 자료구조가 다음과 같을 때, 노드를 가리키는 포인터형 변수 current가 가리키는 노드를 삭제하는 함수 delNode(Node *head, Node *current)를 작성하시오.</p> <p>(폴이)</p> <pre> void delNode(Node *head, Node *current){ Node *p; (p = head;) if(p == NULL) { printf("ERROR!! NOT FOUND..\n"); return; } // p를 사용하여 current가 가리키는 노드를 찾는다. while(p->next != head){ if(p->next == current){ // 찾았으면 리스트에서 삭제한다. (p->next = current->next;) free(current); return; } //다음 노드로 이동하며 찾는다. else (p = p->next;) } //current와 head가 동일한 노드를 가리키는 경우 if(p->next == current){ if(head->next == head){ (head = NULL;) }else { (p->next = current->next;) (head = head->next;) } free(current); return; }else { /* current가 가리키는 노드가 리스트에 존재하지 않는 경우 */ printf("ERROR!! NOT FOUND..\n"); return; } } } </pre>			

게임공학과 전공기초지식 16주			
내용	연결리스트에서의 정렬	선수 연관 과목	프로그래밍
<p>연결리스트에서의 정렬은 배열에서의 정렬과 알고리즘 상 큰 차이가 없다. 그러나 연결리스트의 포인터에 대한 갱신 연산을 수행하여야 하므로 연결리스트의 구조와 포인터 연산에 대한 이해를 높이는 데 큰 도움이 된다.</p> <p>본 실습에서는 배열에 저장된 데이터와 연결리스트에 저장된 데이터에 대하여 각각 정렬을 수행하고, 이 때 소요되는 시간을 비교함으로써 각 데이터 구조에 대한 이해를 높이도록 한다.</p>			
전공 응용 내용	배열과 연결리스트를 이용한 정렬의 비교	전공 연관 과목	3학년 알고리즘
<p>(예제) 배열 Sort 수행 시간과 리스트 Sort 수행 시간을 표시하고 비교한다. (폴이)</p> <pre> void sortarr(void) { long cnt, cn, c; clock_t start, end; struct UNIT temp; start=clock(); for(c=0; c<MAX; c=c*3+1); for(; c>0; c=c/3) { for(cn=c+1; cn<MAX; cn++) { temp=U[cn]; cnt=cn; while(cnt>c && U[cnt-c].hp > temp.hp) { U[cnt]=U[cnt-c]; cnt=cnt-c; } U[cnt]=temp; } } end=clock(); printf("%.3f초!\n", (double)(end-start) /CLOCKS_PER_SEC); system("pause"); } </pre>			

게임공학과 전공기초지식 16주			
전공 응용 내용	배열과 연결리스트를 이용한 정렬의 비교	전공 연관 과목	3학년 알고리즘
<pre> void sortlinked(void) { clock_t start, end; struct node *shead; struct node *temp, *stemp, *now; shead=(struct node *)malloc(sizeof(struct node)); shead->next=NULL; now=shead; start=clock(); while(head->next!=NULL) { stemp=head; temp=stemp; while(stemp->next!=NULL) { if(stemp->hp > stemp->next->hp && temp->next->hp > stemp->next->hp) temp=stemp; stemp=stemp->next; } now->next=temp->next; temp->next=temp->next->next; now=now->next; now->next=NULL; } head->next=shead->next; free(shead); end=clock(); printf("%.3f초!!\n", (double)(end-start) / CLOCKS_PER_SEC); system("pause"); } </pre>			

게임공학과 전공기초지식 17주

내용	스택의 개요 및 연산	선수 연관 과목	프로그래밍
<p>스택이란 여러 개의 데이터 항목들이 일정하게 서로 나열된 자료 구조로, 한쪽 끝에서만 삽입, 삭제할 수 있는 구조이다. 즉, 만일 원소 A, B, C, D, E가 순서대로 스택에 저장되어 있을 때에 삭제 연산을 수행하면 가장 나중에 저장된 E가 가장 먼저 삭제된다. 이와 같이 스택은 가장 나중에 삽입된 원소가 가장 먼저 삭제되는 특징을 가진다. 이러한 특징을 LIFO(Last-In-First-Out)이라고 한다.</p> <div style="text-align: center;">  <p>스택의 구조</p> </div> <p>스택에 대한 연산은 크게 다음의 4가지로 구성된다.</p> <ul style="list-style-type: none"> ● push : 스택에 새로운 데이터 항목을 삽입하는 연산 ● pop : 스택으로부터 데이터 항목을 삭제하는 연산 ● overflow check : 스택이 가득 차 있는지 여부에 대한 검사 ● underflow check : 스택이 비어있는지 여부에 대한 검사 <p>push 연산에 대한 알고리즘은 다음과 같다.</p> <p>단계 1. 스택이 가득 차 있는지 여부를 검사한다.</p> <p>단계 2. stack pointer의 값을 1증가 시킨다.</p> <p>단계 3. stack pointer의 위치에 데이터 항목을 저장한다.</p> <p>pop연산에 대한 알고리즘은 다음과 같다.</p> <p>단계 1. 스택이 비어 있는지 여부를 검사한다.</p> <p>단계 2. stack pointer의 위치에 저장되어 있는 값을 삭제한다.</p> <p>단계 3. stack pointer의 값을 1만큼 감소 시킨다</p>			

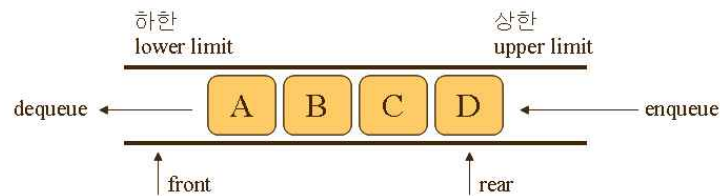
게임공학과 전공기초지식 17주			
전공 응용 내용	스택의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 양방향 연결리스트를 사용하여 스택을 구현하고자 한다. 다음과 같은 구조의 노드를 사용한다고 할 때에 push()와 pop()을 구현하라.</p> <pre> typedef struct _NODE{ char data; struct _NODE *post; //뒤 struct _NODE *pre; //앞 } NODE; </pre> <p>(풀이)</p> <pre> void push(NODE *p, char ch){ p->post= tail=(NODE *)malloc(sizeof(NODE)); tail->data=ch; tail->post=NULL; tail->pre=p; } char pop(){ char a; a = tail->data; tail = tail->pre; return (a); } </pre> <p>(예제) 배열을 이용하여 스택의 push연산과 pop연산을 구현하라.</p> <p>(풀이)</p> <pre> void push(char data){ if (top==MAX){ //스택이 가득 찼는지를 검사 printf("Stack Overflow !\n"); }else{ stack[top]=data; //삽입 top++; //포인터를 증가시킨다 } } char pop(){ if (top < 0){ printf("\nStack Underflow !\n"); }else return stack[--top]; } </pre>			

게임공학과 전공기초지식 18주

내용	큐의 개요 및 연산(1)	선수 연관 과목	프로그래밍
-----------	---------------	-----------------	-------

큐는 한쪽 끝에서 데이터가 삽입되고, 반대쪽 끝에서 데이터가 삭제되는 순서 리스트이다. 즉, 만일 원소 A, B, C, D, E가 순서대로 큐에 삽입되어 있을 때에 삭제 연산을 수행하면 가장 먼저 저장된 A가 가장 먼저 삭제된다. 이와 같이 큐는 가장 처음에 삽입된 원소가 가장 먼저 삭제되는 특징을 가진다. 이러한 특징을 FIFO(First-In-First-Out)이라고 한다.

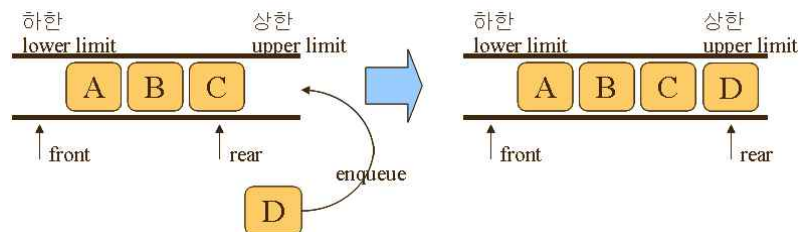
그림은 큐의 구조를 도식화 하여 나타낸 것이다. 그림에서 front는 삭제가 일어나는 곳을 의미하고, rear는 삽입이 일어나는 곳을 의미한다.



선형 큐의 구조

큐에 대한 연산은 크게 다음의 4가지로 구성된다.

- enqueue : 큐에 새로운 데이터 항목을 삽입하는 연산
- dequeue : 큐로부터 데이터 항목을 삭제하는 연산
- full check : 큐가 가득 차 있는지 여부에 대한 검사
- empty check : 큐가 비어있는지 여부에 대한 검사



enqueue 연산이 수행되는 예

게임공학과 전공기초지식 18주			
전공 응용 내용	큐에서 enqueue연산의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 큐에서 enqueue연산의 알고리즘을 기술하라.</p> <p>(풀이)</p> <p>단계 1. 큐가 가득 차 있는지 여부를 검사한다.</p> <p>단계 2. rear pointer의 값을 1증가 시킨다.</p> <p>단계 3. rear pointer의 위치에 데이터 항목을 저장한다.</p> <p>(예제) 단방향 리스트를 사용하여 큐를 구현하고자 한다. 다음과 같은 구조의 노드를 사용한다고 할 때에 enqueue를 구현하라.</p> <pre> typedef struct _node{ int a; struct _node *next; } NODE; </pre> <p>(풀이)</p> <pre> void enqueue(NODE *p, int a) { NODE *temp; while(p->next->a!=NULL) p=p->next; temp=(struct node*)malloc(sizeof(node)); temp->a=a; p->next=temp; temp->next=head; } </pre>			

게임공학과 전공기초지식 19주

내용	큐의 개요 및 연산(2)	선수 연관 과목	프로그래밍
<p>dequeue 연산에 대한 알고리즘은 다음과 같다.</p> <p>단계 1. 큐가 비어 있는지 여부를 검사한다.</p> <p>단계 2. front pointer의 값을 1증가 시킨다.</p> <p>단계 3. front pointer의 위치에서 데이터 항목을 삭제한다.</p> <p>그림은 dequeue연산이 수행되었을 때 큐에 나타나는 변화를 도식화 한 것이다. 그림에서 큐에 4개의 원소가 저장되어 있는 상태에서 dequeue 연산을 수행하면 front pointer가 1만큼 증가하고, front pointer가 가리키는 위치에 저장되어 있는 데이터 원소 'A'가 삭제되는 것을 알 수 있다.</p> <div style="text-align: center;"> </div> <p style="text-align: center;">큐에서 dequeue 연산이 수행되는 예</p> <p>환형큐는 큐의 양 끝 부분이 연결되어 둥근 모양으로 이루어진 큐를 의미한다. 환형 큐의 경우에도 선형 큐와 마찬가지로 put 연산과 get 연산이 있다. 그러나 환형 큐의 경우에는 큐의 끝 부분이 연결되어 있으므로 추가적인 처리가 필요하다.</p>			
전공 응용 내용	큐에서의 dequeue 연산과 환형큐에서의 연산	전공 연관 과목	3학년 알고리즘
<p>(예제) 단방향 리스트를 사용하여 큐를 구현하고자 한다. 다음과 같은 구조의 노드를 사용한다고 할 때에 dequeue를 구현하라.</p> <pre> typedef struct _node{ int a; struct _node *next; } NODE; </pre>			

게임공학과 전공기초지식 19주			
전공 응용 내용	큐에서의 dequeue 연산과 환형큐에서의 연산	전공 연관 과목	3학년 알고리즘
<p>(풀이)</p> <pre> int get(NODE *p) { int temp; NODE *pp; if(p->next->a==NULL){ printf("Data is Empty\n"); return NULL; } } </pre> <p>(예제) 배열로 환형큐를 구성할 때 enqueue 알고리즘을 기술하라.</p> <p>(풀이)</p> <p>단계 1. 큐가 가득 차 있는지 여부를 검사한다.</p> <p>단계 2. rear pointer의 값을 1증가 시킨다.</p> <p>단계 3. rear == upper limit이면 rear pointer를 reset한다.</p> <p>단계 4. rear pointer의 위치에 데이터 항목을 저장한다.</p> <p>(예제) 배열로 환형큐를 구성할 때 dequeue 알고리즘을 기술하라.</p> <p>(풀이)</p> <p>단계 1. 큐가 비어 있는지 여부를 검사한다.</p> <p>단계 2. front pointer의 값을 1증가 시킨다.</p> <p>단계 3. front pointer > upper limit이면, front pointer를 reset 한다.</p> <p>단계 4. front pointer의 위치에서 데이터 항목을 삭제한다.</p>			

게임공학과 전공기초지식 20주			
내용	이진탐색 트리의 개요	선수 연관 과목	프로그래밍
<p>트리는 하나 이상의 노드로 구성된 유한 집합으로서 다음의 조건을 만족한다.</p> <ol style="list-style-type: none"> 1) 루트라는 특별한 노드가 존재한다. 2) 나머지 노드들은 다시 각각이 트리이면서 교차하지 않는 분리집합 $T_1, T_2, \dots, T_N (N>0)$으로 분할된다. 이들을 루트의 서브트리라고 한다. <p>트리에 대한 연산은 크게 검색, 삽입, 삭제 연산이 있다. 이러한 연산들은 주로 재귀함수를 이용하여 구현된다. 본 절에서는 트리의 모든 노드가 하나 또는 두개의 서브트리만을 가지는 이진 탐색 트리를 구현해 보기로 하자.</p> <p>이진 탐색 트리는 다음과 같은 조건을 만족하는 이진트리를 의미한다.</p> <ol style="list-style-type: none"> 1) 모든 노드는 트리 내에서 유일한 키(key)를 가진다. 2) 어떤 노드의 키는 그 노드의 왼쪽 서브트리에 있는 모든 키보다 크다. 3) 어떤 노드의 키는 그 노드의 오른쪽 서브트리에 있는 모든 키보다 작다. <p>이러한 조건을 준수하여 데이터를 삽입, 삭제, 검색함으로써 데이터 관리의 효율성을 높일 수 있다.</p>			
전공 응용 내용	이진탐색트리의 생성	전공 연관 과목	3학년 알고리즘
<p>(예제) 스타크래프트 유닛들에 대한 데이터가 Starc_Unit.txt에 저장되어 있다. 이 데이터를 읽어 이진탐색트리를 구성하라.</p> <p>(폴이)</p> <pre> #define MAX 15 typedef struct _tree { char c; long cnt; int hp; int attack; int def; int range; int mana; struct _tree *left, *right; } TREE; typedef struct _node { char c; long cnt; int hp; int attack; int def; int range; int mana; struct _node *next; } NODE; </pre>			

게임공학과 전공기초지식 20주			
전공 응용 내용	이진탐색트리의 생성	전공 연관 과목	3학년 알고리즘
<pre> typedef struct _unit { char c; long cnt; int hp; int attack; int def; int range; int mana; } UNIT; TREE *Root; NODE *head; UNIT *unit; char dummy; int find; void openfile_tree(void) { FILE *f; long cnt; f=fopen("Starc_Unit.txt","rt"); for(cnt=0; cnt<MAX; cnt++) { TREE *New; New=(TREE *)malloc(sizeof(TREE)); fscanf(f,"%c %ld %d %d %d %d %d\n", &(New->c), &(New->cnt), &(New->hp), &(New->attack), &(New->def), &(New->range), &(New->mana)); New->left=NULL; New->right=NULL; if(cnt==0) Root=New; else insert_tree(&New); } fclose(f); } void insert_tree(TREE **data) { TREE *t, *p; t=Root; p=NULL; while(t!=NULL) { p=t; if(t->hp>(*data)->hp) t=t->left; else t=t->right; } if(p!=NULL) { if(p->hp>(*data)->hp) p->left=*data; else p->right=*data; } } </pre>			

게임공학과 전공기초지식 21주			
내용	이진탐색트리에서의 연산(1)	선수 연관 과목	프로그래밍
<p>이진 탐색트리에 저장된 데이터에 대한 연산으로는 데이터의 검색, 삽입, 삭제, 트리의 운행 등이 있다. 이러한 내용은 본 실습 교재에서 설명하기에는 분량 상 어려움이 있으므로 관련 교재를 참고하도록 한다.</p>			
전공 응용 내용	이진탐색트리에서의 검색 및 운행 연산	전공 연관 과목	3학년 알고리즘
<p>(예제) 스택크래프트 유닛의 데이터가 저장된 이진탐색트리에서 원하는 체력을 가진 유닛의 정보를 검색하는 함수를 작성하라.</p> <p>(풀이)</p> <pre> void search_tree(TREE *root ,int key) { if(root==NULL) return; else { if(root->hp==key) { printf("%3c %8ld %6d %6d %6d %6d %6dWn", root->c, root->cnt, root->hp, root->attack, root->def, root->range, root->mana); find=1; } if(root->hp>key) search_tree(root->left, key); else search_tree(root->right, key); } } </pre>			

게임공학과 전공기초지식 21주			
전공 응용 내용	이진탐색트리에서의 검색 및 운행 연산	전공 연관 과목	3학년 알고리즘
<p>(예제) 스타크래프트 유닛의 데이터가 저장된 이진탐색트리에서 체력이 작은 데이터부터 순서대로 출력하는 함수를 작성하시오.</p> <p>(풀이)</p> <pre>void sort_tree(TREE *root) { if(root==NULL) return; else { sort_tree(root->left); printf("%3c %8ld %6d %6d %6d %6d %6dWn", root->c, root->cnt, root->hp, root->attack, root->def, root->range, root->mana); sort_tree(root->right); } }</pre> <p>(예제) 스타크래프트 유닛의 데이터가 저장된 이진탐색트리에서 체력이 큰 데이터부터 순서대로 출력하는 함수를 작성하시오.</p> <p>(풀이)</p> <pre>void sort_tree2(TREE *root) { if(root==NULL) return; else { sort_tree2(root->right); printf("%3c %8ld %6d %6d %6d %6d %6dWn", root->c, root->cnt, root->hp, root->attack, root->def, root->range, root->mana); sort_tree2(root->left); } }</pre>			

게임공학과 전공기초지식 22주			
내용	이진탐색트리에서의 연산(2)	선수 연관 과목	프로그래밍
<p>이진 탐색트리에 저장된 데이터에 대한 연산으로는 데이터의 검색, 삽입, 삭제, 트리의 운행 등이 있다. 이러한 내용은 본 실습 교재에서 설명하기에는 분량 상 어려움이 있으므로 관련 교재를 참고하도록 한다.</p>			
전공 응용 내용	이진탐색트리에서의 삭제	전공 연관 과목	3학년 알고리즘
<p>(예제) 스택크래프트 유닛의 데이터가 저장된 이진탐색트리에서 원하는 체력을 가진 유닛의 정보를 삭제하는 함수를 작성하라.</p> <p>(풀이)</p> <pre> int delete_tree(TREE **root ,int key) { TREE *t, *p, *child, *subc, *subc_p; t=*root; p=NULL; while(t!=NULL && t->hp!=key) { p=t; if(t->hp>key) t=t->left; else t=t->right; } if(t==NULL) return 0; if(t->left==NULL && t->right==NULL) { if(p!=NULL) { if(p->left==t) p->left=NULL; else p->right=NULL; } else *root=NULL; } } </pre>			

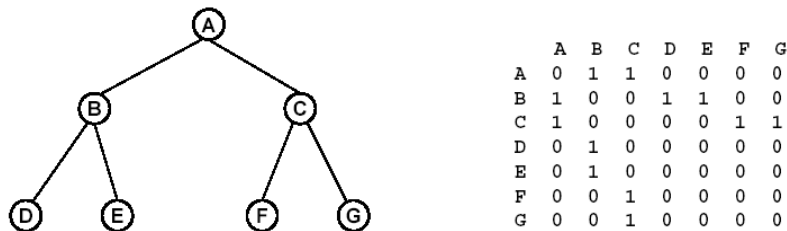
게임공학과 전공기초지식 22주			
전공 응용 내용	이진탐색트리에서의 삭제	전공 연관 과목	3학년 알고리즘
<pre> else if(t->left==NULL t->right==NULL) { if(t->left!=NULL) child=t->left; else child=t->right; if(p!=NULL) { if(p->left==t) p->left=child; else p->right=child; } } else { subc_p=t; subc=t->right; while(subc->left!=NULL) { subc_p=subc; subc=subc->left; } if(subc_p->left==subc) subc_p->left=subc->right; else subc_p->right=subc->right; t->c=subc->c; t->cnt=subc->cnt; t->hp=subc->hp; t->attack=subc->attack; t->def=subc->def; t->range=subc->range; t->mana=subc->mana; t=subc; } find=1; free(t); return 1; } </pre>			

게임공학과 전공기초지식 23주

내용	그래프의 개념 및 운행(1)	선수 연관 과목	프로그래밍
-----------	-----------------	-----------------	-------

그래프는 유한개의 노드와 두개의 노드를 연결하는 유한개의 간선(edge)로 구성된 집합이다. 그래프를 표현하는 방법으로는 인접행렬을 이용하는 방법과 인접리스트를 이용하는 방법이 있다.

n 개의 점을 갖는 그래프의 인접행렬은 $n \times n$ 행렬로 표시되고, 노드 V_i 에서 V_j 사이에 간선이 존재하면 인접행렬 (i, j) 의 값은 1이고, 존재하지 않으면 0이 된다. $n \times n$ 인접행렬의 경우 $n(n-1)$ 개의 간선을 표현할 수 있으나, 대부분의 경우 실제 그래프에서 간선의 개수는 이보다 훨씬 적기 때문에 대부분의 행렬은 0으로 채워지게 되어 기억공간의 낭비가 심한 단점이 있다. 그림은 이진 트리 형태의 그래프를 인접행렬로 표현한 예이다.



인접행렬을 사용하여 그래프를 표현한 예

그래프의 각 노드에는 데이터가 저장된다. 그러므로 간선을 따라 그래프의 각 노드를 액세스하여 데이터를 수집하는 작업이 필요하다. 이러한 작업을 그래프의 운행이라 한다. 그래프의 운행 방법에는 크게 깊이 우선 탐색과 너비 우선 탐색이 있다.

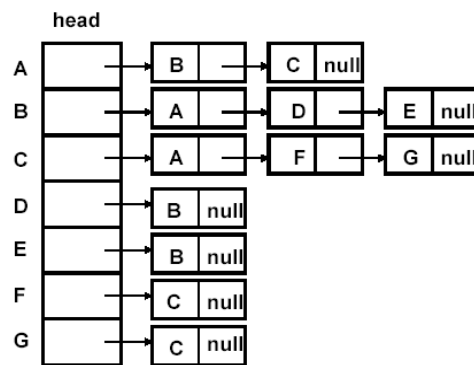
깊이 우선 탐색은 현재 노드로부터 시작하여 아직 방문이 되지 않은 노드를 향해 나아가되 더 이상 갈 곳이 없을 때에는 백 트래킹하는 방법이다. 즉, 시작 노드에서 간선으로 연결된 노드들 중에서 아직 방문하지 않은 노드를 선택하여 다시 깊이 우선 탐색을 적용한다.

게임공학과 전공기초지식 23주			
전공 응용 내용	그래프의 운행방법의 구현(1)	전공 연관 과목	3학년 알고리즘
<p>(예제) 그래프 운행 알고리즘을 구현하기 위해 인접리스트를 위한 구조체를 다음과 같이 정의하자. 이 때 주어진 노드에서 시작하여 깊이 우선 탐색을 하는 함수를 작성하라.</p> <pre> struct adj_node { int node; struct adj_node *next; }; struct adj_list { int tag; struct adj_node *adj; }alist[MAX_NODES]; (폴이) void dfs(int node) { struct adj_node *p; printf("%d ", node); alist[node].tag =VISITED; p=alist[node].adj; while(p != NULL) { if(alist[p->node].tag != VISITED) dfs(p->node); p = p->next; } } </pre>			

게임공학과 전공기초지식 24주

내용	그래프의 개념 및 운행(2)	선수 연관 과목	프로그래밍
-----------	-----------------	-----------------	-------

인접리스트는 인접행렬에서의 n 개의 행을 n 개의 리스트로 표현한 것이다. 다음 그림은 지난번에 보았던 이진트리 형태의 그래프를 인접리스트로 표현한 것이다.



인접리스트를 사용하여 그래프를 표현한 예

인접 리스트는 간선이 없는 경우에는 표현 자체가 되지 않으므로 일반적인 경우 인접 행렬에 비하여 저장공간의 낭비가 적다. 그러나 각 노드를 연결하는 포인터의 저장 공간으로 인하여 표현하려는 그래프가 완전 그래프에 가까울수록 저장공간의 낭비가 심해지는 문제가 있다.

너비 우선 탐색은 시작 노드를 방문한 후 시작 노드와 간선으로 연결된 노드들을 차례대로 모두 방문한다. 더 이상 방문할 노드가 없으면 시작노드와 인접한 노드들에 인접한 노드들을 다시 차례로 방문한다. 이러한 과정을 모든 노드가 방문될 때까지 반복한다.

게임공학과 전공기초지식 24주			
전공 응용 내용	그래프의 운행방법의 구현(2)	전공 연관 과목	3학년 알고리즘
<p>(예제) 주어진 노드에서 시작하여 너비 우선 탐색을 하는 함수를 작성하라.</p> <p>(풀이)</p> <pre> void bfs(int node) { int node; struct adj_node *tmp; put(node); // 큐에 삽입한다. alist[node].tag = VISITED; while((node = get()) != QUEUE_EMPTY){ printf("%d ", node); tmp =alist[node].adj; while(tmp != NULL) { if(alist[tmp->node].tag != VISITED) { put(tmp->node); alist[tmp->node].tag =VISITED); } tmp =tmp->next; } } } </pre>			

게임공학과 전공기초지식 25주			
내용	정렬의 개념 및 선택 정렬	선수 연관 과목	프로그래밍
<p>정렬이란 임의의 순서대로 배열되어 있는 자료의 집합을 일정한 순서대로 재배열하는 과정을 의미한다. 정렬은 크게 내부정렬과 외부정렬로 구분된다.</p> <p>내부정렬은 정렬하려는 데이터를 모두 메모리에 올려놓고 정렬하는 방법으로서 속도가 매우 빠른 장점이 있다. 그러나 모든 데이터가 메모리에 상주하여야 하므로 정렬하려는 데이터가 매우 많은 경우에는 사용이 어렵다. 대표적인 내부정렬 방식에는 선택정렬, 퀵정렬, 삽입정렬, 병합정렬 등이 있다.</p> <p>선택정렬은 가장 큰 것을 선택하여 가장 마지막 것과 교환하는 방식으로 정렬을 진행하는 방법이다. 다음은 선택정렬의 수행 단계에 따라 데이터가 정렬되는 과정을 보인 것이다.</p> <p style="text-align: center;">90 78 100 30 55</p> <p style="text-align: center;">30 78 100 90 55</p> <p style="text-align: center;">30 55 100 90 78</p> <p style="text-align: center;">30 55 78 90 100</p>			
전공 응용 내용	선택정렬의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 선택정렬 알고리즘을 기술하라.</p> <p>(풀이)</p> <p>단계 1. 가장 작은(또는 큰) 값을 찾는다.</p> <p>단계 2. 해당 값을 첫번째 위치의 값과 교환한다.</p> <p>단계 3. 두번째 작은(또는 큰) 값을 찾는다.</p> <p>단계 4. 해당 값을 두번째 위치의 값과 교환한다.</p> <p>단계 5. 전체가 정렬될 때까지 단계 1부터 단계 4까지의 작업을 반복한다.</p>			

게임공학과 전공기초지식 25주			
전공 응용 내용	선택정렬의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 주어진 배열에 저장된 데이터를 선택정렬을 사용하여 정렬하는 함수를 작성하라.</p> <p>(풀이)</p> <pre> void selection_sort(int x[], int lim){ int i; int eff_size; int maxpos; int tmp; for (eff_size = lim; eff_size > 1; eff_size--) { maxpos=eff_size-1; for (i = 0; i < eff_size; i++) maxpos = x[i] > x[maxpos] ? i : maxpos; tmp = x[maxpos]; x[maxpos] = x[eff_size - 1]; x[eff_size - 1] = tmp; } } </pre>			

게임공학과 전공기초지식 26주			
내용	버블 정렬과 삽입 정렬	선수 연관 과목	프로그래밍
<p>버블정렬은 인접한 배열의 요소를 비교, 교환하여 최대(또는 최소)값을 배열의 제일 뒤로 보내는 작업을 반복함으로써 정렬하는 방법이다. 다음은 버블정렬 알고리즘을 기술한 것이다.</p> <p>단계 1. 인접한 두 값을 비교한다. 단계 2. 앞의 값이 뒤의 값보다 크면(또는 작으면) 두 값의 위치를 교환한다. 단계 3. 그렇지 않으면 다음 인접한 두 값에 대하여 단계 1부터 단계 2까지의 작업을 반복한다. 단계 4. 모든 키의 비교가 끝나면 한 단계가 종료한다. 단계 5. 한 단계가 종료하면 다시 처음부터 모든 값이 정렬될 때까지 단계 1부터 단계 4까지의 작업을 반복한다.</p> <p>삽입정렬은 이미 정렬된 데이터들이 있을 때 새로운 데이터를 적절한 위치에 삽입하는 방식으로 정렬을 진행하는 방법이다. 다음은 삽입정렬의 알고리즘을 기술한 것이다.</p> <p>단계 1. 정렬된 데이터들에서 새로운 데이터가 삽입될 위치를 찾는다. 단계 2. 찾은 위치에 삽입할 공간을 확보한다. 단계 3. 새로운 데이터를 삽입한다. 단계 4. 모든 데이터에 대해서 단계 1부터 단계 3까지의 작업을 반복한다.</p>			
전공 응용 내용	버블정렬과 삽입정렬의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 주어진 배열에 저장된 데이터를 버블정렬을 사용하여 정렬하는 함수를 작성하라.</p> <p>(풀이)</p> <pre>void bubble_sort(int a[], int n){ int i, j, temp; for (i = 0; i < n - 1; ++i) for (j = n - 1; j > i; --j) if (a[j - 1] > a[j]){ temp = a[j-1]; a[j-1] = a[j]; a[j] = temp; } }</pre>			

게임공학과 전공기초지식 26주			
전공 응용 내용	버블정렬과 삽입정렬의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 주어진 배열에 저장된 데이터를 버블정렬을 사용하여 정렬하는 함수를 작성하라.</p> <p>(풀이)</p> <pre>void bubble_sort(int a[], int n){ int i, j, temp; for (i = 0; i < n - 1; ++i) for (j = n - 1; j > i; --j) if (a[j - 1] > a[j]){ temp = a[j-1]; a[j-1] = a[j]; a[j] = temp; } }</pre> <p>(예제) 다음의 데이터를 선택정렬에 따라 정렬하는 과정을 보이라.</p> <p>90 78 100 30 55</p> <p>(풀이)</p> <p>90 78 100 30 55</p> <p>78 90 100 30 55</p> <p>78 90 100 30 55</p> <p>30 78 90 100 55</p> <p>30 55 78 90 100</p> <p>(예제) 주어진 배열에 저장된 데이터를 삽입정렬을 사용하여 정렬하는 함수를 작성하라.</p> <p>(풀이)</p> <pre>void insertion_sort(int item[], int n) { int i,j,next; for(i=1;i<n;i++){ next=item[i]; for(j=i; item[j-1] > next && j>=0; j--){ item[j]=item[j-1]; } item[j]=next; } }</pre>			

게임공학과 전공기초지식 27주			
내용	퀵 정렬	선수 연관 과목	프로그래밍
<p>퀵정렬은 가장 빠르고 많이 쓰이는 방법으로서, 정렬할 데이터를 기준 값보다 큰 값들과 작은 값들로 분할하고, 각각의 집합에 대하여 다시 퀵정렬을 적용하는 방식으로 정렬을 진행하는 방법이다. 다음은 퀵정렬의 알고리즘을 기술한 것이다.</p> <p>퀵정렬알고리즘(a, n) // n은 데이터의 개수</p> <p>단계 1. 만약 $n > 1$이면</p> <p> 단계 1.1 n개의 요소를 갖는 배열 a에 대하여 축값보다 작은 수들은 배열의 앞쪽에, 큰 수들은 배열의 뒤쪽에 위치시키고 축값은 그 경계에 위치시킨 후 축값의 위치를 mid에 넣는다. 축값은 임의로 정한다.</p> <p> 단계 1.2 퀵정렬 알고리즘(a, mid)을 호출한다.</p> <p> 단계 1.3 퀵정렬 알고리즘(a+mid+1, n-mid-1)을 호출한다.</p>			
전공 응용 내용	퀵정렬의 구현	전공 연관 과목	3학년 알고리즘
<p>(예제) 주어진 배열에 저장된 데이터를 퀵정렬을 사용하여 정렬하는 함수를 작성하라.</p> <p>(풀이)</p> <pre>void quicksort(element list[], int left, int right){ int pivot; int i, j ; element temp; if (left<right) { i = left ; j = right + 1; pivot = list[left].key;</pre>			

게임공학과 전공기초지식 27주			
전공 응용 내용	퀵정렬의 구현	전공 연관 과목	3학년 알고리즘
<pre> /* 왼쪽과 오른쪽 서브리스트로부터 키를 검색하여, 오른쪽과 왼쪽 경계가 교차하거나 만날 때까지 순서에 벗어난 원소들을 교환한다 */ do { do{ i++; }while (list[i].key<pivot); do{ j--; }while (list[j].key>pivot); if(i<j){ temp = list[i]; list[i] = list[j]; list[j] = temp; } } while (i<j); temp = list[left]; list[left] = list[right]; list[right] = temp; quicksort(list, left, j-1); quicksort(list, j+1, right); } } </pre>			

게임공학과 전공기초지식 28주			
내용	자료구조를 이용한 게임 - 오목(1)	선수연관 과목	프로그래밍
<p>본 실습에서는 지금까지 습득한 자료구조 지식을 바탕으로 간단한 텍스트 게임을 개발한다. 본 실습에서 개발할 게임은 대표적인 보드 게임의 하나인 “오목”이다.</p> <p>오목에 대한 구현을 단계적으로 세분화 하여 몇 시간에 걸쳐 오목을 완성한다. 자료구조의 여러 가지 지식을 실제 게임에 적용해 봄으로써 지금까지 습득한 지식의 활용방법을 익힐 수 있을 것이다.</p>			
전공 응용 내용	오목의 구현(1)-상황판단	전공연관 과목	3학년 알고리즘
<p>(예제) 오목에서 가로, 세로, 대각선 방향으로 연속적으로 놓인 바둑알의 개수가 가장 많은 경우에 대해 바둑알이 놓인 자리의 X, Y 좌표를 표시하시오.</p> <p>(폴이)</p> <pre> #define WHITE -1 #define BLACK 1 #define MAX 19 struct DOL { int x; int y; int dirx; int diry; int result; } ; struct SAVE { int x; int y; struct SAVE * next; }; struct SAVE *head; struct SAVE *Nhead; void initnode(void); int input_pan(int (*pan)[MAX], struct DOL *dol); void print_pan(int (*pan)[MAX], struct DOL *dol); int rule_main(int (*pan)[MAX], struct DOL *dol); int rule_line(int (*pan)[MAX], int i, int j, int x, int y); void save(int x, int y); void Delete(void); int turn; char dummy; int result; int PAN[MAX][MAX]={0}; struct DOL arr[4]; int main(void) </pre>			

게임공학과 전공기초지식 28주			
전공 응용 내용	오목의 구현(1)-상황판단	전공연관 과목	3학년 알고리즘
<pre> { print_pan(PAN, arr); result=input_pan(PAN, arr); turn=turn*(-1); } print_pan(PAN, arr); if(turn==BLACK) printf("○ 승리!\n"); else printf("● 승리!\n"); Delete(); free(Nhead); free(head); return 0; } void print_pan(int (*pan)[MAX], struct DOL *dol) { int i, j; int x=0, y=0; int temp; int SUM_w=0, SUM_b=0; system("cls"); for(i=0; i<19; i++) { for(j=0; j<19; j++) { if(pan[i][j]==WHITE) { printf("○"); SUM_w++; } else if(pan[i][j]==BLACK) { printf("●"); SUM_b++; } else if(i==0) { if(j==0) printf("┌"); else if(j==18) printf("┐"); else printf("┴"); } else if(i==18) { if(j==0) printf("└"); else if(j==18) printf("┘"); else printf("┬"); } else { if(j==0) printf("├"); else if(j==18) printf("┤"); else printf("┼"); } } } } </pre>			

게임공학과 전공기초지식 28주			
전공 응용 내용	오목의 구현(1)-상황판단	전공연관 과목	3학년 알고리즘
<pre> } printf("\n"); } temp=turn; turn=WHITE; printf("○ 돌: %d개 최대 %d연속", SUM_w, rule_main(pan,dol)); printf("\n가로 최대: %d", dol[0].result); for(i=0;i<dol[0].result;i++) { printf("(%d ,%d) ",(dol[0].x)+ x+ 1, (dol[0].y)+ y+ 1); x=x+ (dol[0].dirx); y=y+ (dol[0].diry); } x=y=0; printf("\n역대각 최대: %d", dol[2].result); for(i=0;i<dol[2].result;i++) { printf("(%d ,%d) ",(dol[2].x)+ x+ 1, (dol[2].y)+ y+ 1); x=x+ (dol[2].dirx); y=y+ (dol[2].diry); } printf("\n"); turn=BLACK; printf("● 돌: %d개 최대 %d연속", SUM_b, rule_main(pan,dol)); printf("\n세로 최대: %d", dol[1].result); for(i=0;i<dol[1].result;i++) { printf("(%d ,%d) ",(dol[1].x)+ x+ 1, (dol[1].y)+ y+ 1); x=x+ (dol[1].dirx); y=y+ (dol[1].diry); } printf("\n대각 최대: %d", dol[3].result); for(i=0;i<dol[3].result;i++) { printf("(%d ,%d) ",(dol[3].x)+ x+ 1, (dol[3].y)+ y+ 1); x=x+ (dol[3].dirx); y=y+ (dol[3].diry); } printf("\n"); turn=temp; } </pre>			

게임공학과 전공기초지식 29주			
내용	자료구조를 이용한 게임 - 오목(2)	선수 연관 과목	프로그래밍
<p>본 실습에서는 지금까지 습득한 자료구조 지식을 바탕으로 간단한 텍스트 게임을 개발한다. 본 실습에서 개발할 게임은 대표적인 보드 게임의 하나인 “오목”이다. 오목에 대한 구현을 단계적으로 세분화 하여 몇 시간에 걸쳐 오목을 완성한다. 자료구조의 여러 가지 지식을 실제 게임에 적용해 봄으로써 지금까지 습득한 지식의 활용방법을 익힐 수 있을 것이다.</p>			
전공 응용 내용	오목의 구현(2) - 승패판정	전공 연관 과목	3학년 알고리즘
<p>(예제) 오목에서 연속으로 5개의 바둑알이 놓인 경우를 탐지하여 승리를 선언하도록 하시오.</p> <p>(풀이)</p> <pre> int rule_line(int (*pan)[MAX], int i, int j, int x, int y) { int xx=x, yy=y; int SUM_user=1; while((i+x)<19 && (j+y)<19 && (i+x)>=0 && (j+y)>=0) { if(pan[i+x][j+y]==pan[i+x-xx][j+y-yy] && pan[i+x][j+y]==turn) SUM_user++; x=x+xx; y=y+yy; if(pan[i+x][j+y]==turn*(-1) pan[i+x][j+y]==0) break; } return SUM_user; } int rule_main(int (*pan)[MAX], struct DOL *dol) { int i, j; result=0; for(i=0; i<19; i++) { for(j=0; j<19; j++) { if(pan[i][j]==turn && pan[i][j]==pan[i][j+1] && pan[i][j]!=pan[i][j-1]) { if(dol[0].result<=rule_line(pan,i,j,0,1)) </pre>			

게임공학과 전공기초지식 29주			
전공 응용 내용	오목의 구현(2) - 승패판정	전공 연관 과목	3학년 알고리즘
<pre> { if(turn==WHITE) { dol[0].x=i; dol[0].y=j; dol[0].dirx=0; dol[0].diry=1; dol[0].result=rule_line(pan,i,j,0,1); } if(dol[0].result>result) result=dol[0].result; } } if(pan[i][j]==turn && pan[i][j]==pan[i+1][j] && pan[i][j]!=pan[i-1][j]) { if(dol[1].result<=rule_line(pan,i,j,1,0)) { if(turn==BLACK) { dol[1].x=i; dol[1].y=j; dol[1].dirx=1; dol[1].diry=0; dol[1].result=rule_line(pan,i,j,1,0); } if(dol[1].result>result) result=dol[1].result; } } if(pan[i][j]==turn && pan[i][j]==pan[i+1][j+1] && pan[i][j]!=pan[i-1][j-1]) { if(dol[2].result<=rule_line(pan,i,j,1,1)) { if(turn==WHITE) { dol[2].x=i; dol[2].y=j; dol[2].dirx=1; dol[2].diry=1; dol[2].result=rule_line(pan,i,j,1,1); } if(dol[2].result>result) result=dol[2].result; } } } if(pan[i][j]==turn && pan[i][j]==pan[i+1][j-1] && pan[i][j]!=pan[i-1][j+1]) </pre>			

게임공학과 전공기초지식 29주			
전공 응용 내용	오목의 구현(2) - 승패판정	전공 연관 과목	3학년 알고리즘
<pre> { if(dol[3].result<=rule_line(pan,i,j,1,-1)) { if(turn==BLACK) { dol[3].x=i; dol[3].y=j; dol[3].dirx=1; dol[3].diry=-1; dol[3].result=rule_line(pan,i,j,1,-1); } if(dol[3].result>result) result=dol[3].result; } } if(pan[i][j]==turn && pan[i][j]==pan[i][j-1] && pan[i][j]!=pan[i][j+1]) { if(result<=rule_line(pan,i,j,0,-1)) result=rule_line(pan,i,j,0,-1); } if(pan[i][j]==turn && pan[i][j]==pan[i-1][j] && pan[i][j]!=pan[i+1][j]) { if(result<=rule_line(pan,i,j,-1,0)) result=rule_line(pan,i,j,-1,0); } if(pan[i][j]==turn && pan[i][j]==pan[i-1][j-1] && pan[i][j]!=pan[i+1][j+1]) { if(result<=rule_line(pan,i,j,-1,-1)) result=rule_line(pan,i,j,-1,-1); } if(pan[i][j]==turn && pan[i][j]==pan[i-1][j+1] && pan[i][j]!=pan[i+1][j-1]) { if(result<=rule_line(pan,i,j,-1,1)) result=rule_line(pan,i,j,-1,1); } } } return result; } </pre>			

게임공학과 전공기초지식 30주			
내용	자료구조를 이용한 게임 - 오목(3)	선수 연관 과목	프로그래밍
<p>본 실습에서는 지금까지 습득한 자료구조 지식을 바탕으로 간단한 텍스트 게임을 개발한다. 본 실습에서 개발할 게임은 대표적인 보드 게임의 하나인 “오목”이다.</p> <p>오목에 대한 구현을 단계적으로 세분화 하여 몇 시간에 걸쳐 오목을 완성한다. 자료구조의 여러 가지 지식을 실제 게임에 적용해 봄으로써 지금까지 습득한 지식의 활용방법을 익힐 수 있을 것이다.</p>			
전공 응용 내용	오목의 구현(3)- 기보화일 만들기	전공 연관 과목	3학년 알고리즘
<p>(예제) 화일 입출력을 이용하여 오목의 기보 파일을 읽어 오목판을 구성하라.</p> <p>(풀이)</p> <pre> void savefile(void){ FILE *f; struct SAVE *temp; f=fopen("gibo.txt","wt"); temp=head; while(temp->next!=NULL) { temp=temp->next; fprintf(f, " %d %d ", temp->x, temp->y); } fclose(f); } void openfile(void){ FILE *f; f=fopen("gibo.txt","rt"); while(!feof(f)) { struct SAVE *New; New=(struct SAVE *)malloc(sizeof(struct SAVE)); fscanf(f, " %d %d ", &New->x, &New->y); New->next=Nhead->next; Nhead->next=New; } fclose(f); } </pre>			

게임공학과 전공기초지식 31주			
내용	자료구조를 이용한 게임 - 오목(4)	선수 연관 과목	프로그래밍
<p>본 실습에서는 지금까지 습득한 자료구조 지식을 바탕으로 간단한 텍스트 게임을 개발한다. 본 실습에서 개발할 게임은 대표적인 보드 게임의 하나인 “오목”이다.</p> <p>오목에 대한 구현을 단계적으로 세분화 하여 몇 시간에 걸쳐 오목을 완성한다. 자료구조의 여러 가지 지식을 실제 게임에 적용해 봄으로써 지금까지 습득한 지식의 활용방법을 익힐 수 있을 것이다.</p>			
전공 응용 내용	오목의 구현(4)- UNDO와 REDO	전공 연관 과목	3학년 알고리즘
<p>(예제) 스택을 이용하여, 오목을 물리거나 원래 상태로 복구 시키는 기능을 구현하시오.</p> <p>(풀이)</p> <pre> void Pan_UNDO(int (*pan)[MAX]) { struct SAVE *del; int i, j; del=head->next; if(del==NULL) { printf("무를것이 없습니다\n"); system("pause"); } else { pan[del->x][del->y]=0; head->next=del->next; del->next=Nhead->next; Nhead->next=del; } } </pre>			

게임공학과 전공기초지식 31주			
전공 응용 내용	오목의 구현(4)- UNDO와 REDO	전공 연관 과목	3학년 알고리즘
<pre> for(i=0; i<MAX; i++) { for(j=0; j<MAX; j++) { if(pan[i][j]==-2) pan[i][j]=0; } } void Pan_REDO(int (*pan)[MAX]) { struct SAVE *del; del=Nhead->next; if(del==NULL) { printf("돌릴것이 없습니다\n"); system("pause"); } else { pan[del->x][del->y]=turn; Nhead->next=del->next; del->next=head->next; head->next=del; } } </pre>			