게임 자료구조와 알고리즘
-CHAPTER4-

SOULSEEK

목차

C스타일 동적할당

1. C스타일 동적할당

- 메모리 공간

⇨ 실행할 프로그램의 명령코드 저장 공간

⇨ 전역변수,Static변수 저장 공간

⇨ 지역변수, 매개변수 저장 공간

⇨ 동적할당 저장공간

1. C스타일 동적할당

- **정적할당**
  -Compile(컴파일)시 할당될 메모리의 크기가 결정
  -Stack,Data 영역에 변수로 할당되는 메모리

- **동적할당**
  -실행 중에 메모리 할당
  -동적할당 메모리는 Heap영역에 할당

- 동적할당 사용 예
  - 컴파일 시점에서 데이터의 크기를 정할 수 없는 경우
  - 정적으로 할당할 경우 반듯이 그 만큼의 메모리를 활용하거나 그 이상일때 난처하거나 낭비가 되는 경우가 많다 특정해서 사용하지 않고 필요할 때 필요한 만큼 사용하기위한 경우

1. C스타일 동적할당

Malloc

- **Heap**영역에 동적으로 공간을 할당 해 주는 함수
- 필요 헤더파일 = **<stdlib.h>**
- 형식 = **(void*)malloc(size_t size);**

  할당 하고싶은 자료형을 **Byte**단위로 인자값을 보낸 후
  할당된 메모리의 시작주소를 해당 자료형으로 형변환 후
  Stack 또는 **Data**영역에서 사용한다.
  ex> int* Num = (int*)malloc(4);

```
#include<stdlib.h>

void main()
{
	int* pNum = (int*)malloc(sizeof(int));

	*pNum = 10;

	cout << "pNum주소 = " << pNum  << ", npNum값 = " << *pNum << endl;
}
```

1. C스타일 동적할당

Heap영역의 특징

- Heap영역은 자료형의 개념이 없어 Byte단위로 정보를 처리한다.
- Heap영역은 일반적인 Stack과 Data영역보다 메모리영역이 넓다.
- 할당된 메모리의 시작주소를 malloc함수를 호출한 지역으로 반환한다.
- 시작주소만 반환 되기 때문에 Stack영역에서 온전히 사용하기 위해 해당 자료형 주소로 형변환을 해야 한다.
- Heap영역에 할당된 메모리는 자동으로 해제되지 않는다.

1. C스타일 동적할당

Free

- **Heap**영역에 할당된 공간을 해제하는 함수
- 필요 헤더파일 = **<stdlib.h>**
- 형식 = **free(void*)**
  Heap영역에 할당된 메모리는 자동으로 해제되지 않는다.
  ex> int* Num = (int*)malloc(4);
  free(Num);

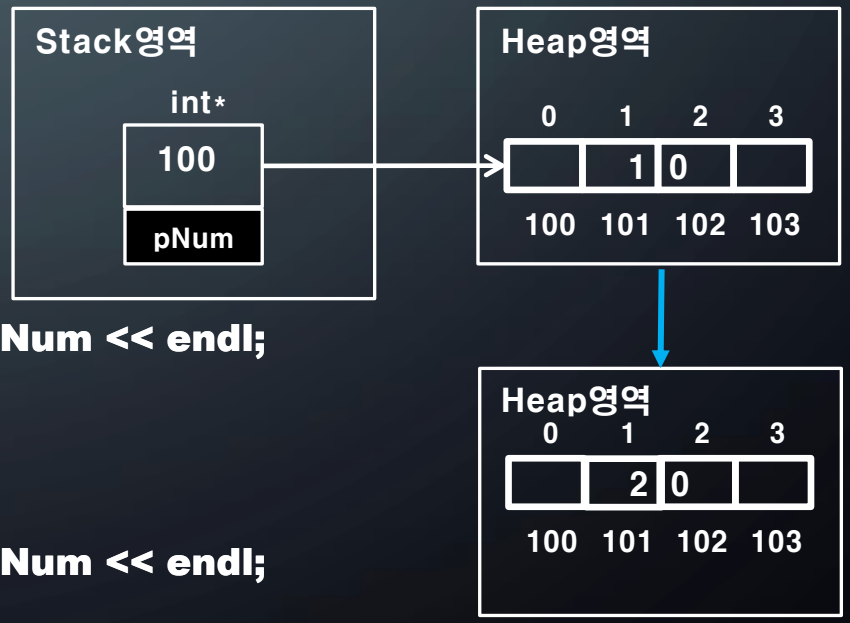
```
#include<stdlib.h>

void main()
{
	int* pNum = (int*)malloc(sizeof(int));
	*pNum = 10;

	cout << "pNum주소 = " << pNum << ", npNum값 = " << *pNum << endl;

	free(pNum);

	*pNum = 20;
	cout << "pNum주소 = " << pNum << ", npNum값 = " << *pNum << endl;
}
```



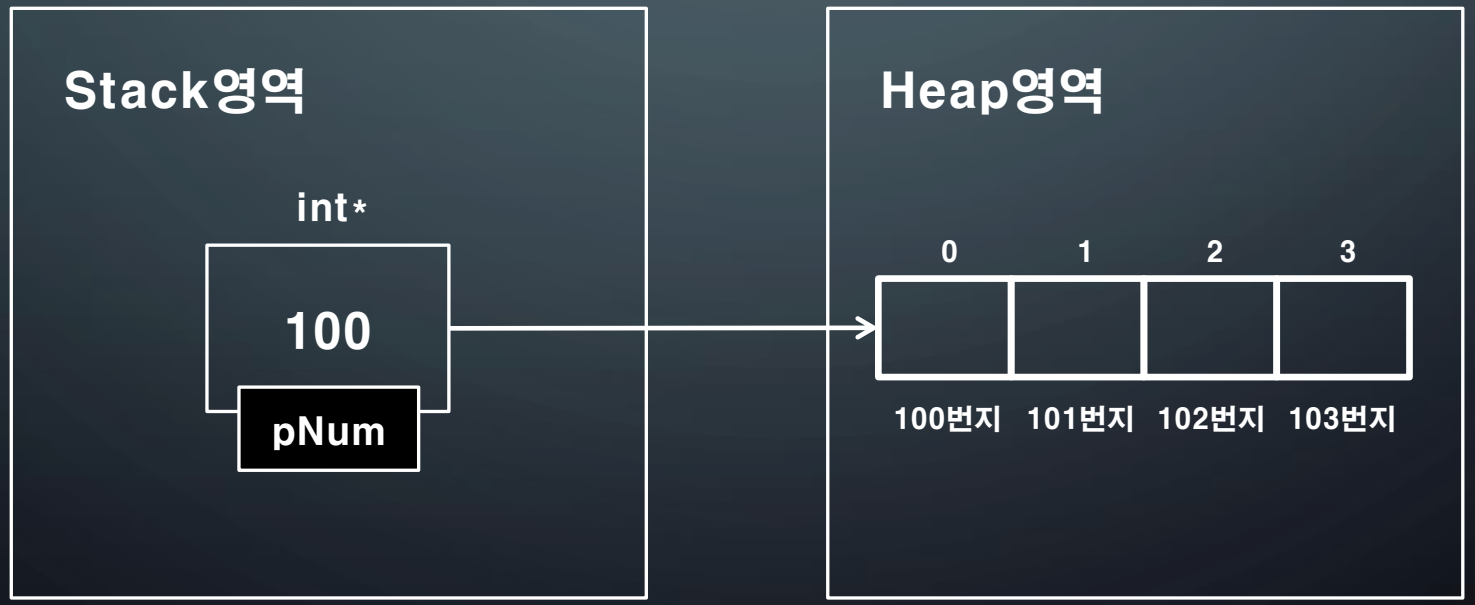
1. C스타일 동적할당

```
#include<stdlib.h>

void main()
{
	int* pNum = (int*)malloc(sizeof(int));
	*pNum = 10;

	cout << "pNum주소 = “ << pNum << “ , npNum값 = “ << *pNum << endl;

	free(pNum);
	pNum= NULL;
}
```



1. C스타일 동적할당

```
#include<stdlib.h>

void main()
{
	int* pNumArr;
	int Size;
	cout << "할당할 공간의 갯수를 입력하시오 : ";
	cin >> Size;
	pNumArr = (int*)malloc(sizeof(int)* Size);

	for (int i = 0; i < Size; i++)
		pNumArr[i] = i + 1;

	for (int i = 0; i < Size; i++)
		cout << i << " = “ << pNumArr[i] << endl;

	free(pNumArr);
	pNumArr = NULL;
}
```

1. C스타일 동적할당

학생 정보를 입력받는 동적할당 응용 코드 참고

C++스타일 동적할당

2. C++스타일 동적할당

new

- **Heap**영역에 동적으로 공간을 할당 해 주는 함수
- 필요 헤더파일 = **<iostream>**
- 형식 = **new DataType**
  new 키워드를 쓴 뒤 할당하고 싶은 자료형을 입력하여 **Heap**영역에서 동적할당을 하고 시작주소를 반환 받는다
  ex> int* Num = new int;

delete

- **Heap**영역에 할당된 공간을 해제하는 함수
- 필요 헤더파일 = **<iostream>**
- 형식 = **delete void***
  Heap영역에 할당된 메모리는 자동으로 해제되지 않는다.
  ex> int* Num = new int;
  delete Num;

2. C++스타일 동적할당

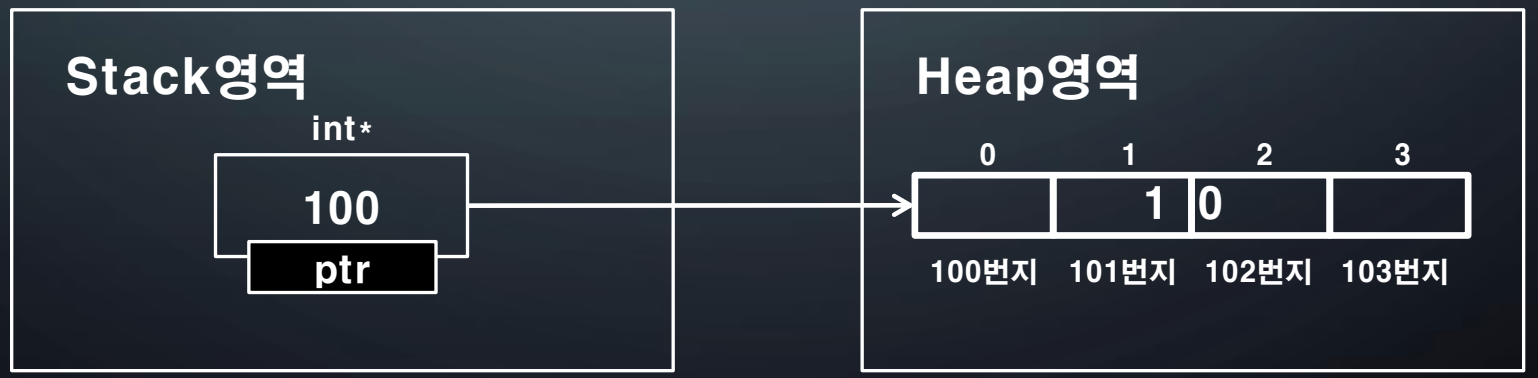
```
#include <iostream>
using namespace std;

void main()
{
    int *ptr;

    ptr = new int;

    *ptr = 10;
    cout << *ptr << endl;
    cout << ptr << endl;

    delete ptr;
}
```



2. C++스타일 동적할당

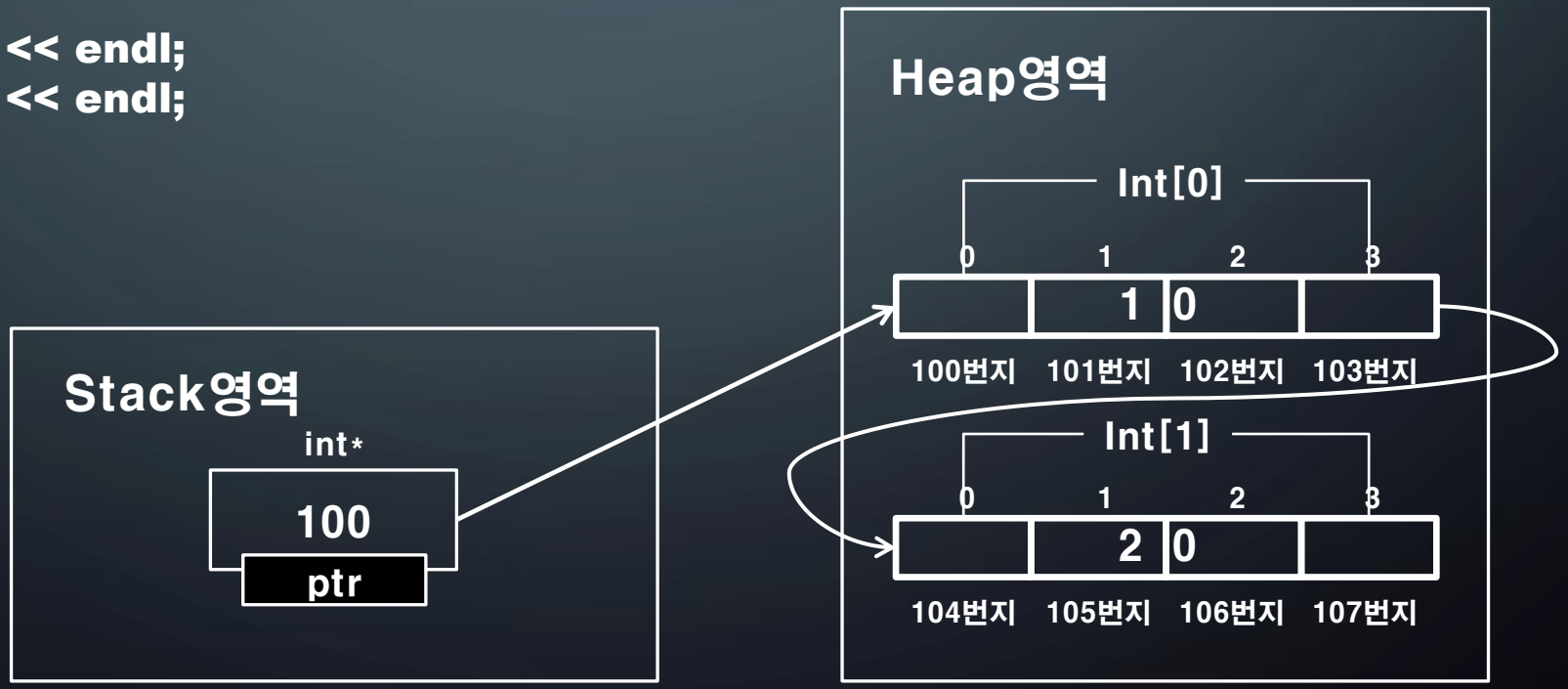
```
#include <iostream>
using namespace std;

void main()
{
    int *ptr;

    ptr = new int[2];

    ptr[0] = 10;
    ptr[1] = 20;
    cout << ptr[0] << endl;
    cout << ptr[1] << endl;

    delete []ptr;
}
```



2. C++스타일 동적할당

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

void main()
{
    int size;
    string* student;
    cout << "등록할 학생수를 입력하시오\n";
    cin >> size;

    student = new string[size];

    for(int i = 0 ; i < size; i++)
    {
      cout << i+1 <<"번 학생 이름 등록";
      cin >> student[i];
    }
    cout << endl << endl << endl;

    for(int i = 0 ; i < size ; i++)
    {
      cout << i+1 << "번학생 :";
      cout << student[i] << endl;
    }

    delete[] student;
}
```

2. C++스타일 동적할당

학습과제

- C스타일의 동적할당 응용코드를 C++형식의 동적할당형식으로 교체해 보자.
- 동적할당을 이용해서 첫번째 수 입력 받은 수만큼 배열을 할당하고 두번째 수 입력 받은 만큼 오름차순, 내림차순 정렬하여 출력해보자.
- Player의 닉네임, 레벨, 랭킹, 전투력을 입력 받는 구조체를 만든 뒤 첫번째로 입력 받은 수 만큼 구조체를 동적 할당하고 수만큼 입력 받은 뒤 입력을 더 이상 할 수 없다고 출력! 이 후 각 입력 받은 수치들을 각 항목별로 선택해서 오름차순으로 출력해보자(닉네임 제외))

문자열 - C스타일

3. 문자열 - C스타일

문자열함수

정의

문자열을 원하는 방법으로 **제어**할 수 있도록 도와주는 함수

필요 헤더파일 = **<string.h>**

- 종류
  - **strlen :** 문자열 길이 반환
  - **strcpy :** 문자열을 복사해주는 함수
  - **strcat :** 문자열을 추가해주는 함수
  - **strcmp :** 문자열을 비교해주는 함수

3. 문자열 - C스타일

strlen – 문자열 길이

```
#include<string.h>

void main()
{
	char str[10] = "Hello";

	cout << str << "%s문자열의 길이 : “ << strlen(str) << endl;
}
```

3. 문자열 - C스타일

strcpy – 문자열 복사

```
#include<string.h>

void main()
{
	char Name[10];
	char My_Name[10] = “SoulSeek";

	strcpy(Name, My_Name);
	cout << "Name : “ << Name << endl;
	cout << "My_Name : “ << My_Name << endl;
}
```

3. 문자열 - C스타일

Strcat – 문자열 합치기

```
#include<string.h>

void main()
{
	char str[10] = "Hello";

	cout << str << endl;

	strcat(str, "^^");

	cout << str << endl;
}
```

3. 문자열 - C스타일

strcmp – 문자열비교 : 첫번째와 두번째의 관계

```
#include<stdio.h>
#include<string.h>

void main()
{
	char str1[10] = "string!!";
	char str2[10] = "string";

	//str10이 str2보다 크므로 1
	printf( " %s == %s : %d\n " , str1, str2, strcmp(str1, str2));
	//abc와 abc가 같으니까 0
	printf( " %s == %s : %d\n " ,  " abc " ,  " abc " , strcmp( " abc " ,  " abc " ));
	//abc보다 def가 크므로 -1
	printf("%s == %s : %d\n", "abc", "def", strcmp("abc", "def"));
}
```

3. 문자열 - C스타일

학습과제

- **strlen, strcpy, strcat, strcmp**의 역할을 하는 함수를 직접 작성하자**.**

문자열 - C++스타일

4. 문자열 – C++스타일

string 자료형

- **C++**에서 사용하는 **문자열 자료형**이다.
- **내부함수**가 다양하게 있어 활용하기 수월하다.
- 필요 헤더파일 **: <string>**

4. 문자열 – C++스타일

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

void main()
{
	string str = "Hello";
	cout << "str = " << str << endl << endl;
	cout << "새로운 문자열 입력 : ";
	cin >> str;
	cout << "새로운 str = " << str << endl;
	return;
}
```

4. 문자열 – C++스타일

얕은 복사/깊은 복사

- 문자열의 **얕은 복사**
  특정 문자의 주소값을 복사한 문자열로 연결된 문자열이 변경되면 다른 문자열도 변경된다.(주소 복사)
- 문자열의 **깊은 복사**
  특정 문자의 메모리를 그대로 복사하여 한 문자열이 변경되더라도 다른 문자열은 변경되지 않는다.
  (메모리 복사)

4. 문자열 – C++스타일

char 배열 얕은복사 – C스타일

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

void main()
{
	char str[6] = "Hello";
	char* tmp = str;
	cout << "str = " << str << endl;
	cout << "tmp = " << tmp << endl;
	strcpy(str, "Bye");
	cout << "str = " << str << endl;
	cout << "tmp = " << tmp << endl;

	return;
}
```

4. 문자열 – C++스타일

char 배열 깊은복사 – C스타일

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

void main()
{
	char str[6] = "Hello";
	char tmp[6];

	for (int i = 0; i < 6; i++)
		tmp[i] = str[i];

	cout << "str = " << str << endl;
	cout << "tmp = " << tmp << endl;

	strcpy(str, "Bye");

	cout << "str = " << str << endl;
	cout << "tmp = " << tmp << endl;

	return;
}
```

4. 문자열 – C++스타일

sting 배열 얕은복사 – C++스타일

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

void main()
{
	string str = "Hello";

	char* arr = (char*)str.c_str();

	cout << "str = " << str << endl;
	cout << "arr = " << arr << endl;

	str = "Bye";

	cout << "str = " << str << endl;
	cout << "arr = " << arr << endl;

	return;
}
```

4. 문자열 – C++스타일

string 배열 깊은복사 – C++스타일

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

void main()
{
	string str1 = "Hello";
	string str2;

	str2 = str1;

	cout << "str1 = " << str1 << endl;
	cout << "str2 = " << str2 << endl;

	str1 = "Bye";

	cout << "str1 = " << str1 << endl;
	cout << "str2 = " << str2 << endl;

	return;
}
```

4. 문자열 – C++스타일

문자열 길이

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

void main()
{
	string s1;
	string s2 = "123";
	string s3 = "Hello";
	string s4 = "안녕하세요";
	cout << "s1 = " << s1.length() << endl;
	cout << "s2 = " << s2.length() << endl;
	cout << "s3 = " << s3.length() << endl;
	cout << "s4 = " << s4.length() << endl;

	//cout << "s4 = " << s4.size() << endl;
	return;
}
```

4. 문자열 – C++스타일

문자열의 비교와 추가

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

void main()
{
	string str;
	while (1)
	{
		system("cls");
		cout << "Very를 입력하시오 : ";
		cin >> str;
		if (str == "Very")
		{
			str += "Good";
			cout << str << endl;
			break;
		}
		cout << "잘못 입력 다시" << endl;
		system("pause");
	}
	return;
}
```

4. 문자열 – C++스타일

문자열 가져오기

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

void main()
{
    string str = "Education is the best provision for old age.\n - Aristotle";
    int index = str.find("provision");
    cout << "Find Provision = " << index << endl;
    cout << str[index] << endl;
    cout << str.substr(index, sizeof("provision"));
    return;
}
```

입출력(I/O) - C스타일

5. 입출력(I/O) - C스타일

- 프로그램 내의 정보를 **외부 파일에** 저장하거나 외부 파일의 정보를 프로그램으로 **불러오는** 방식

- 필요 헤더파일 **<stdio.h>**

- 사용 함수
  -fprintf : 파일의 내용 출력
  -fscanf : 파일의 내용 입력

- 입출력 옵션
  -"w"모드 **:** 해당 이름의 파일 **덮어쓰기**상태로 연다
  (파일이 없을 경우 새로만든다**)**
  (파일에 내용이 있을경우 전부 지운 후 다시 작성 한다**.)**

  -"a"모드 **:** 해당 이름의 파일 **추가**상태로 연다
  (파일이 없을 경우 새로만든다**)**
  (파일에 내용이 있을 경우 마지막 내용 뒤에 추가한다**.)**

  -"r"모드 **:** 해당 이름의 파일 **읽기**상태로 연다
  (파일이 없을 경우 **NULL**을 반환한다**.)**

5. 입출력(I/O) - C스타일

fprintf(“w”옵션) - 덮어쓰기

```
#include<stdio.h>

void main()
{
	FILE* f = fopen("Test.txt", "w");
	int Num = 123;
	fprintf(f, "덮어 쓰기 모드  %d", Num);
	fclose(f);
}
```

5. 입출력(I/O) - C스타일

fprintf(“a”옵션) - 추가하기

```
#include<stdio.h>

void main()
{
	FILE* f = fopen("Test.txt", "a");
	int Num = 123;
	fprintf(f, "추가 모드  %d", Num);
	fclose(f);
}
```

5. 입출력(I/O) - C스타일

fscanf(“r”옵션) – 띄어쓰기나 엔터문자 단위로

```
#include<stdio.h>

void main()
{
	FILE* f = fopen("Test.txt", “w");
	int Num;
	fprintf(f,"1 2 3 4");
	fclose(f);
	f = fopen("Test.txt", "r");

	if (f == NULL)
		printf("해당 이름의 파일이 없습니다.");
	else
	{
		while (!feof(f))
		{
			fscanf(f, "%d", &Num);
			printf("%d", Num);
		}
		fclose(f);
	}

}
```

5. 입출력(I/O) - C스타일

fprintf("r"옵션) – 띄어쓰기나 엔터문자 단위로

```
#include<stdio.h>

typedef struct people
{
	char Name[10];
	int Age;
	char PhonNumber[20];
}People;

void main()
{
	People P1 = { "A",20,"010-1234-5678" };
	FILE* f = fopen("People.txt", "w");
	fprintf(f,"%s %d %s",P1.Name,P1.Age,P1.PhonNumber);
	fclose(f);
	f = fopen("People.txt", "r");
	if (f == NULL)
		printf("해당 이름의 파일이 없습니다.");
	else
	{
		fscanf(f, "%s", P1.Name);
		fscanf(f, "%d", &P1.Age);
		fscanf(f, "%s", P1.PhonNumber);
		printf("이름 : %s \n나이 : %d \n", P1.Name, P1.Age);
		printf("휴대폰번호 : %s\n", P1.PhonNumber);
	}
	fclose(f);
}
```

5. 입출력(I/O) - C스타일

fgets(“r”옵션) – 엔터문자 단위로

```
#include<stdio.h>

typedef struct people
{
	char Name[10];
	int Age;
	char PhonNumber[20];
}People;

void main()
{
	People P1 = { "A",20,"010-1234-5678" };
	FILE* f = fopen("People.txt", "w");
	char buf[256];
	fprintf(f, "이름 : %s나이 : %d\n", P1.Name, P1.Age);
	fprintf(f, "휴대폰 번호 : %s", P1.PhonNumber);
	fclose(f);
	f = fopen("People.txt", "r");
	if (f == NULL)
		printf("해당 이름의 파일이 없습니다.");
	else
	{
		while (!feof(f))
		{
			fgets(buf, sizeof(buf), f);
			printf("%s\n", buf);
		}
		fclose(f);
	}
}
```

5. 입출력(I/O) - C스타일

fread("r"옵션) – 전체 내용을 읽는다.

```
#include<stdio.h>

typedef struct people
{
	char Name[10];
	int Age;
	char PhonNumber[20];
}People;

void main()
{
	People P1 = { "A",20,"010-1234-5678" };
	FILE* f = fopen("People.txt", "w");
	char buf[256];

	//memset(buf, 0, sizeof(256)); // string.h 필요
	//ZeroMemory(buf, 256); // Windows.h 필요

	fprintf(f, "이름 : %s나이 : %d\n", P1.Name, P1.Age);
	fprintf(f, "휴대폰 번호 : %s", P1.PhonNumber);
	fclose(f);
	f = fopen("People.txt", "r");
	if (f == NULL)
		printf("해당 이름의 파일이 없습니다.");
	else
	{
		fread(buf, sizeof(buf),1,f);
		printf("%s\n", buf);
	}
	fclose(f);
```

입출력(I/O) – C++스타일

6. 입출력(I/O) – C++스타일

- 프로그램 내의 정보를 **외부 파일에** 저장하거나 외부 파일의 정보를 프로그램으로 **불러오는** 방식

- 필요 헤더파일 **<fstream>**

- 사용 함수
  -<< : 파일의 내용 출력
  ->> : 파일의 내용 입력

- 입출력 변수
  -ofstream : 쓰기모드 형태로 파일정보를 담는 자료형
  (파일이 없을 경우 새로만든다**)**
  (default값 덮어쓰기모드**, ios:app :** 추가모드**)**
  -ifstream : 읽기모드 형태로 파일정보를 담는 자료형
  (파일이 없을 경우 **NULL**을 반환한다**.)**

6. 입출력(I/O) – C++스타일

ofstream

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

void main()
{
	ofstream save;
	save.open("test.txt");
	if (save.is_open())
	{
		save << "이건 파일 입출력" << endl << "입니다.";
		save.close();
	}

}
```

6. 입출력(I/O) – C++스타일

ofstream

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
void main()
{
	ofstream save;
	save.open("test.txt",ios::app);
	if (save.is_open())
	{
		save << "\n이건 파일 입출력추가모드";
		save.close();
	}
}
```

6. 입출력(I/O) – C++스타일

ifstream - >> 띄워쓰기 또는 개행단위로

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
void main()
{
	ofstream save;
	save.open("test.txt");
	if (save.is_open())
	{
		save << "이건 파일 입출력";
		save.close();
	}

	ifstream load;
	string str;
	string tmp;
	load.open("test.txt");
	while (!load.eof())
	{
		load >> tmp;
		str += tmp;
		str += " ";
	}
	cout << str;
}
```

6. 입출력(I/O) – C++스타일

ifstream - >> 띄워쓰기 또는 개행단위로

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
void main()
{
	ofstream save;
	save.open("test.txt");
	if (save.is_open())
	{
		save << "이건 파일 입출력";
		save.close();
	}

	ifstream load;
	string str;
	string tmp;
	load.open("test.txt");

	while (!load.eof())
	{
		getline(load, str);
		cout << str<<endl;
	}
}
```

6. 입출력(I/O) – C++스타일

학습과제

- 지뢰찾기에 타이머를 부착해서 종료 난위도와 시간을 Save하자.
- 고급 난위도로 진행해서 고급 난위도가 종료되면 기록되어 있는 시간을 Load해서 비교한 뒤 빨리 종료한 시간으로 다시 저장해보자.
- 미로찾기 진행중에 지정한 문자를 입력 받으면 위치를 저장하자.(재저장시 덮어씌우기)
- 미로찾기를 시작 할 때 로드버튼을 이용해 저장정보를 활용해서 시작위치를 셋팅하자.