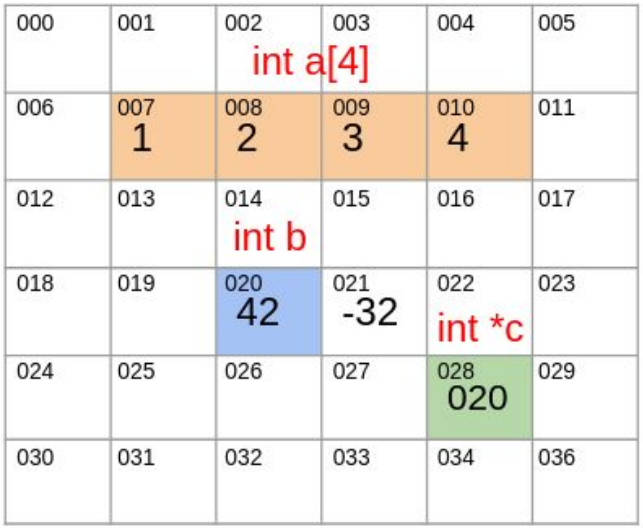
Numerical Analysis

C Programming 기초 복습

C Pointer

메모리, 배열, 변수, 포인터?

- **메모리**: 데이터를 저장할 수 있는 공간의 집합
  - 각 공간은 고유의 주소를 가지고 있음
- **변수**: 메모리상에 할당받은 특정 공간
  - b 는 020주소의 공간을 할당받음
- **배열**: 메모리상에 할당받은 연속된 공간
  - a는 007~010주소의 공간을 할당받음
- **포인터**: 주소를 저장하기 위한 용도의 변수
  - c는 028주소의 공간을 할당받음

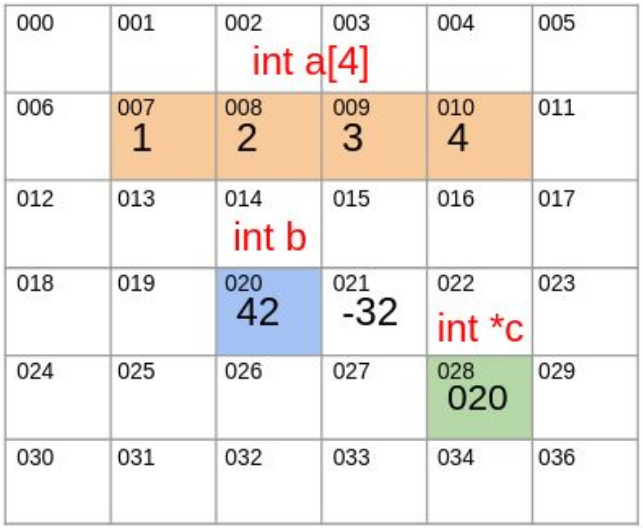


Memory

C Pointer

포인터 관련 연산자

- &(): 변수를 인자로 받아 변수 할당받은 주소를 반환
  - &b → 014
  - &c → 028
- *(): 주소를 인자로 받아 주소 저장된 값을 반환
  - *(008) → 2
  - *c → *(020) → 42
- ()[ ]: 주소와 offset을 인자로 받아 주소 + offset에 저장된 값을 반
  - a[2] → (007)[2] → *(007 + 002) → *(009) → 3
  - c[1] → (020)[1] → *(020 + 001) → *(021) → −32



Memory

C Pointer 사용 예제

여러 값을 반환하는 함수

```
void machar(int *ibeta, int *it, int *irnd, int *ngrd, int *machep, int *negep,
        int *iexp, int *minexp, int *maxexp, float *eps, float *epsneg,
        float *xmin, float *xmax)
```

All of `machar`'s arguments are returned values. Here is what they mean:

- `ibeta` (called B in §1.3) is the radix in which numbers are represented, almost always 2, but occasionally 16, or even 10.
- `it` is the number of base-`ibeta` digits in the floating-point mantissa M (see Figure 1.3.1).
- `machep` is the exponent of the smallest (most negative) power of `ibeta` that, added to 1.0, gives something different from 1.0.
- `eps` is the floating-point number $\texttt{ibeta}^{\texttt{machep}}$, loosely referred to as the "floating-point precision."
- `negep` is the exponent of the smallest power of `ibeta` that, subtracted from 1.0, gives something different from 1.0.
- `epsneg` is $\texttt{ibeta}^{\texttt{negep}}$, another way of defining floating-point precision. Not infrequently `epsneg` is 0.5 times `eps`; occasionally `eps` and `epsneg` are equal.
- `iexp` is the number of bits in the exponent (including its sign or bias).
- `minexp` is the smallest (most negative) power of `ibeta` consistent with there being no leading zeros in the mantissa.
- `xmin` is the floating-point number $\texttt{ibeta}^{\texttt{minexp}}$, generally the smallest (in magnitude) useable floating value.
- `maxexp` is the smallest (positive) power of `ibeta` that causes overflow.
- `xmax` is $(1-\texttt{epsneg}) \times \texttt{ibeta}^{\texttt{maxexp}}$, generally the largest (in magnitude) useable floating value.

C Function Pointer

Function Pointer

- 함수를 가리키는 포인터
- returnType (*ptr) (arguments)의 형식으로 구성됨

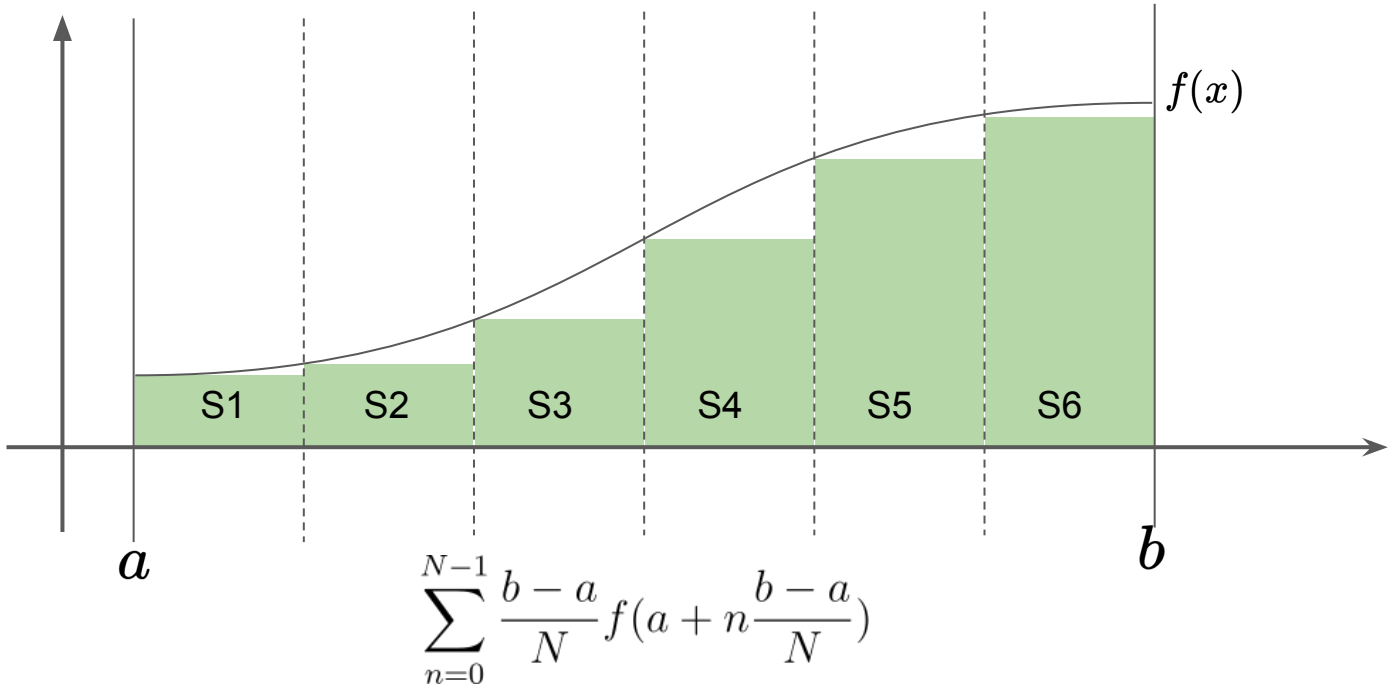
Function과 Function Pointer의 차이

- float add(float a, float b);
- float (*funcPtr)(float, float);

Return Type	Name	Arguments
float	add	(float, float)
float	(*funcPtr)	(float, float)

C Function Pointer 사용 예제

함수의 리만합(Riemann sum) 계산



$$\sum_{n=0}^{N-1} \frac{b-a}{N} f(a + n\frac{b-a}{N})$$

C Function Pointer 사용 예제

```c
#include <stdio.h>
#include <math.h>

float linear(float x) { return 2.f*x; }
float wave(float x) { return sinf(x); }

float RS(float (*func)(float), float begin, float end, int step){
        float stepSize = (end - begin)/(float)step;
        float result = 0.f;

        int i;
        for(i = 0; i<step; i++)
                result += stepSize*func(begin + (float)i*stepSize);

        return result;
}

int main(){

        float result0 = RS(linear, 0.f, 10.f, 100);
        printf("Result of 2x [0, 10]: %0.10f\n", result0);

        float result1 = RS(wave, 0.f, 3.141592f, 100);
        printf("Result of sin(x) [0, PI]: %0.10f\n", result1);

        float result2 = RS(wave, -3.141592f, 3.141592f, 100);
        printf("Result of sin(x) [-PI, PI]: %0.10f\n", result2);
        return 0;
}
```

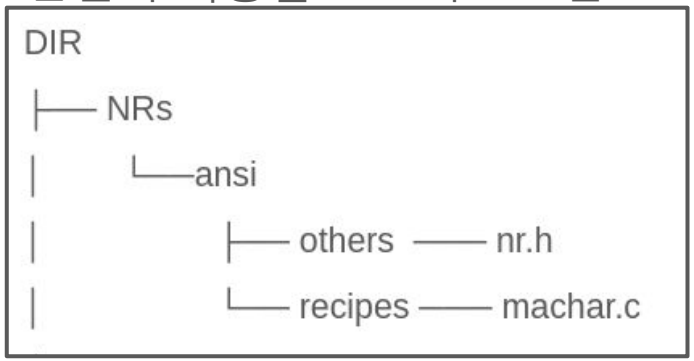
- 왼쪽의 예제는 함수를 인자로 받아 해당 함수의 리만합을 구하는 예시임
- 함수 포인터를 사용하면 함수를 인자로 받아 처리할 수 있음
- 수치해석에서 다루는 기법의 상당수는 특정 함수의 해나 극대, 극소점을 찾는 방법임
- 앞으로 하게될 과제들에 함수 포인터를 활용하는 연습이 필요함

```
Result of 2x [0, 10]: 99.0000076294
Result of sin(x) [0, PI]: 1.9998352528
Result of sin(x) [-PI, PI]: -0.0000000503
```

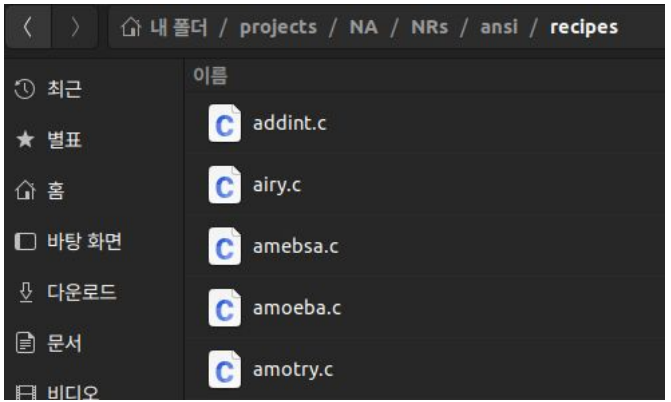
기초 Numerical Recipes 사용법 (Linux/MacOS/WSL)

Numerical Recipes 함수의 사용

- Numerical Recipes의 모는 함수는 동일 이름의 *.c파일에 작성되어 있음 (예 machar.c)
- 그 밖의 자료형과 함수의 인터페이스는 모두 nr.h에 선언되어 있음
- 본인이 작성할 코드에 nr.h를 include해야 원하는 함수를 호출해서 사용가능



Numerical Recipes 소스 구성



기초 Numerical Recipes 사용법 (Linux/MacOS/WSL)

machar() 함수를 사용하여 Machine Accuracy 출력하기

```c
#include <stdio.h>
#include "nr.h"

void get_eps(float *eps){
        *eps = 0.f;
        //Your Implementation
}

int main(){
        int ibeta, it, irnd, ngrd, machep, negep, iexp, minexp, maxexp;
        float eps, epsneg, xmin, xmax;

        machar(&ibeta, &it, &irnd, &ngrd, &machep, &negep, &iexp, &minexp, &maxexp,
                        &eps, &epsneg, &xmin, &xmax);
        printf("Machine Accuracy (machar): \t%0.20f\n", eps);

        get_eps(&eps);
        printf("Machine Accuracy (get_eps): \t%0.20f\n", eps);

        return 0;
}
```

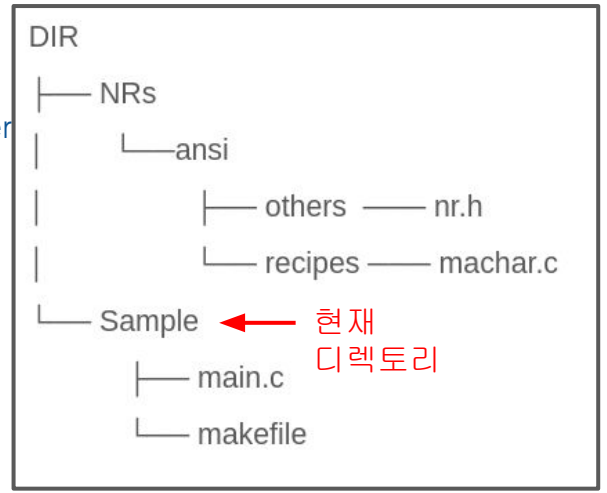
sampleMachar/main.c

기초 Numerical Recipes 사용법 (Linux/MacOS/WSL)

machar() 함수를 사용하여 Machine Accuracy 출력하기

cc/gcc를 사용한 C 컴파일

- cc -o runMachAr main.c ../NRs/ansi/recipes/machar.c -I ../NRs/ansi/other
  - main.c와 machar.c 소스파일을 컴파일하여 runMachAr 실행파일 생성
  - -o 옵션을 통해 결과파일의 이름 지정 (-o name)
  - -I 옵션을 통해 헤더파일을 찾을 경로 추가 (-I directory)



디렉토리 구성

Makefile을 통한 C 컴파일

- [target name] : dependencies

  compile command

터미널에서 **make**명령어 실행시 **dependencies**에 기재된 파일이 수정될 경우 **compile command**가 실행됨

```
runMachAr : main.c ../NRs/ansi/recipes/machar.c ../NRs/ansi/other/nr.h
	cc -o runMachAr main.c ../NRs/ansi/recipes/machar.c -I ../NRs/ansi/other
```

기초 Numerical Recipes 사용법 (Linux/MacOS/WSL)

machar() 함수를 사용하여 Machine Accuracy 출력하기

- 터미널에 make명령어 실행 (Makefile의 기재된 대로 컴파일 수행)
- 생성된 runMachAr 실행파일 실행

```
junyoung@mrlab:~/projects/NA/SampleMachar$ make; ./runMachAr
cc -o runMachAr main.c ../NRs/ansi/recipes/machar.c -I ../NRs/ansi/other
In file included from main.c:2:
../NRs/ansi/other/nr.h:183:7: warning: conflicting types for built-in function ‘fmin’; exp
ected ‘double(double,  double)’ [-Wbuiltin-declaration-mismatch]
  183 | float fmin(float x[]);
      |       ^~~~
../NRs/ansi/other/nr.h:1:1: note: ‘fmin’ is declared in header ‘<math.h>’
  +++ |+#include <math.h>
    1 | #ifndef _NR_H_
Machine Accuracy (machar):      0.00000011920928955078
Machine Accuracy (get_eps:      0.00000000000000000000
```