

NA-HW #01

<machar() & get_eps()>

2015004693_양상현

[Method 1] Using 'machar()' in NR.

먼저 'machar.c'에 이미 구현되어 있는 machar() 함수를 그대로 소스 코드에 복사하여 선언해주었다. 이후 main() 함수 안에서 machar() 함수를 실행하고 machar() 함수 안에 들어가는 인자들을 함수 실행 후에 콘솔창에 출력해주어 결과를 확인하였다.

machar() 함수를 실행할 때 함수안에 들어가는 인자들은 단순히 main() 함수 안에서 각각의 변수형에 맞게 선언만 해주고 초기화하는 과정을 생략하고 진행했다.

기존의 machar() 함수는 'float' 타입으로 되어있어서 'double' 타입에 대한 결과를 얻기 위해 별개의 기존의 machar() 함수를 macharFloat() 으로 선언하고, 기존의 machar() 함수의 인자 부분과 내부에 선언된 변수형 중 'float'으로 선언된것을 모두 'double'로 바꾼 macharDouble() 함수를 분리하여 선언하였다.

macharFloat() 함수와 macharDouble() 함수를 모두 초기화되지 않은 변수들을 인자로 실행을 해보았을 때, 3쪽에 결과 사진처럼 각각 다른 결과가 나오게 되는 것을 알 수 있고, 인자로 들어가는 'eps'가 Machine Accuracy(Epsilon)를 의미하고, 'it'가 $1+2^{(-n)}=1$ 을 만족하는 n 값이라는 것을 알 수 있다.

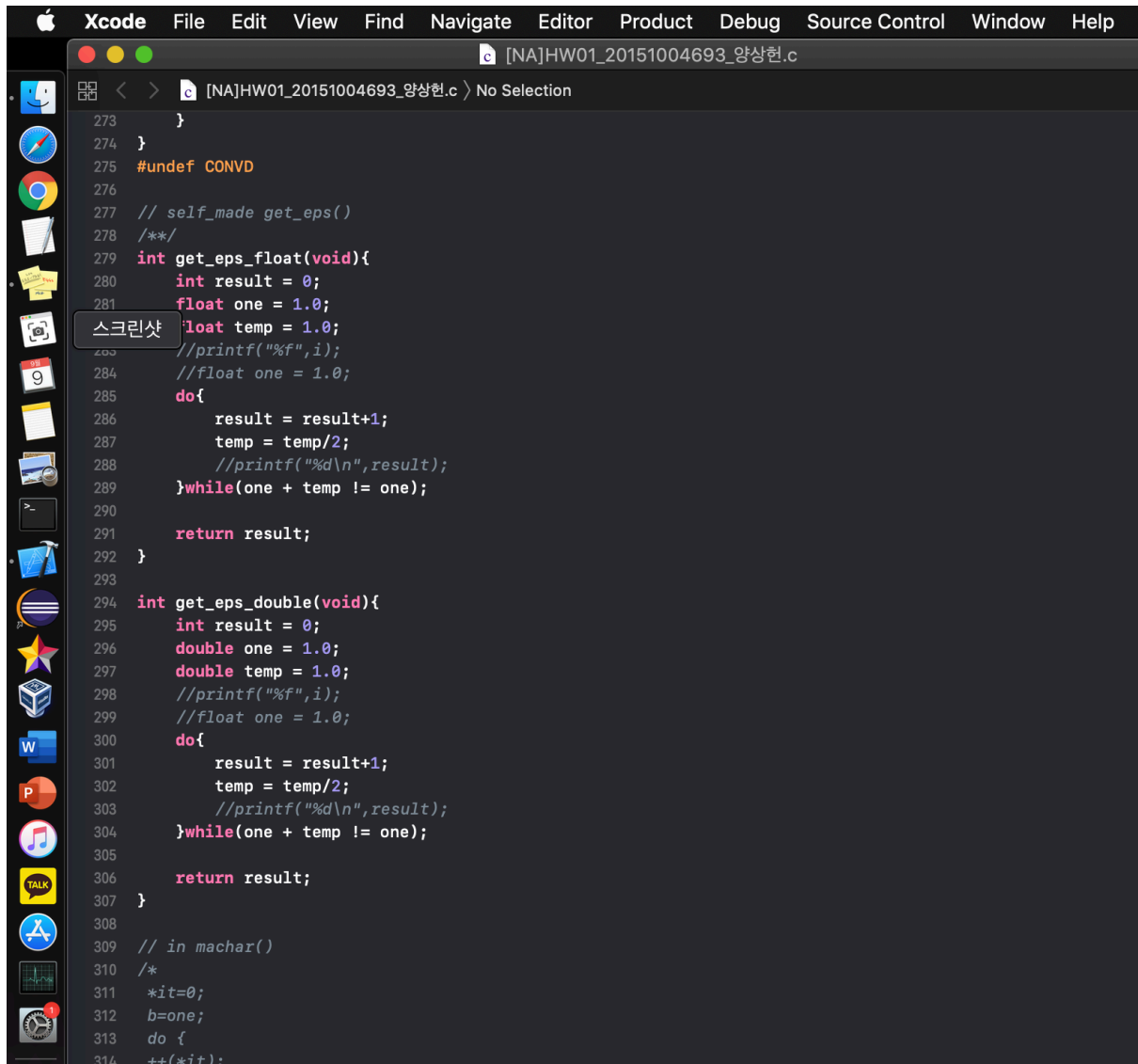
[Method 2] Using self-made 'get_eps()' function.

$1+2^{(-n)}=1$ 을 만족하는 n 을 찾는 함수 인 get_eps()는 2 쪽에 사진과 같이 제작했다.

$1+2^{(-n)}=1$ 식을 이용하여 반복문 안에서 식이 성립할 때까지 1/2 을 계속해서 곱하고 1/2 이 곱해질 때마다 위 식에서 의 'n'을 의미하는 'result' 변수가 1 씩 증가하는 방식으로 제작했다. 함수가 종료되면 $1+2^{(-n)}=1$ 를 만족하는 'n' 값이 들어있는 'result'를 반환하게 된다.

'float'형과 'double'형을 구분하기 위해 함수를 따로 만들어 변수를 각각 'float' 형과 'double'형으로 선언 후 결과를 비교할 수 있도록 하였다.

<get_eps()함수>



```
273 }
274 }
275 #undef CONV
276
277 // self_made get_eps()
278 /**
279 int get_eps_float(void){
280     int result = 0;
281     float one = 1.0;
282     float temp = 1.0;
283     //printf("%f",i);
284     //float one = 1.0;
285     do{
286         result = result+1;
287         temp = temp/2;
288         //printf("%d\n",result);
289     }while(one + temp != one);
290
291     return result;
292 }
293
294 int get_eps_double(void){
295     int result = 0;
296     double one = 1.0;
297     double temp = 1.0;
298     //printf("%f",i);
299     //float one = 1.0;
300     do{
301         result = result+1;
302         temp = temp/2;
303         //printf("%d\n",result);
304     }while(one + temp != one);
305
306     return result;
307 }
308
309 // in machar()
310 /**
311 *it=0;
312 b=one;
313 do {
314     ++(*it);
```

[Results of Method 1&2]

```
mac Finished running NA_NR_practice : NA_NR_practice {}
NA_NR_practice > NA_NR_practice > main.c > main()
359 printf("maxexp = %d\n",maxexpD);
360 printf("eps = %12.6g\n",epsD);
361 printf("epsneg = %12.6g\n",epsnegD);
362 printf("xmin = %12.6g\n",xminD);
363 printf("xmax = %12.6g\n",xmaxD);
364 //printf("//-----//\n");
365
366 //result of self-made get_eps()
367 printf("//-----'N' by 'get_eps()'-----//\n");

//---for Float---//
ibeta = 2
it = 24
irnd = 5
ngrd = 0
machep = -23
negep = -24
iexp = 8
minexp = -126
maxexp = 128
eps = 1.19209e-07
epsneg = 5.96046e-08
xmin = 1.17549e-38
xmax = 3.40282e+38
//---for Double---//
ibeta = 2
it = 53
irnd = 5
ngrd = 0
machep = -52
negep = -53
iexp = 11
minexp = -1022
maxexp = 1024
eps = 2.22045e-16
epsneg = 1.11022e-16
xmin = 2.22507e-308
xmax = 1.79769e+308
//-----'N' by 'get_eps()'-----//
'N' epsilon for float using get_eps_float(): 24
'N' epsilon for float using get_eps_Double(): 53
//-----//
Program ended with exit code: 0

All Output Filter
```

출력된 결과를 확인했을 때, 'macharFloat()' 에서의 'it'와 'get_eps_float()' 의 결과 값이 24 로 같고, 'macharDouble()' 에서의 'it'와 'get_eps_double()' 의 결과 값이 53 으로 같다. 또한 'float'의 Machine Accuracy(Epsilon)는 1.19209e-07 이고, 'double'의 Machine Accuracy(Epsilon)는 2.22045e-16 임을 알 수 있다.