Algorithm

Assignment 4

물리학과 20182326 이선민

problem1

```
[~/2022_1/Algorithm/assignment4]$ ./a.out
fibonacci 5 : 5
fibonacci 10 : 55
```

- The array of data was declared as a global variable. The fibonacci recursive function was stored in data, and if there was a value in data, the recursive function was not called and the value was immediately taken from the data and returned.

```c
#include <stdio.h>

int data[100];

int fibo(int x)
{
        if (x <= 2)
                return 1;
        if (data[x] != 0)
                return data[x];
        else
        {
                data[x] = fibo(x-1) + fibo(x-2);
                return data[x];
        }
}

int main()
{
        printf("fibonacci 5 : %d\n", fibo(5));
        printf("fibonacci 10 : %d\n", fibo(10));
}
```

Problem2 -code

```
58 int main()
59 {
60         int **order;
61         int p[4] = {5,3,7,10};
62         int number = 3;
63         int result;
64         int a[100][100];
65         int b[100][100];
66         int c[100][100];
67         int tmp[100][100];
68         int final[100][100];
69
70         int i,j,k;
71         printf("Number of array: %d\n", number);
72
73         for (i = 0; i < 4; i++)
74                 printf("p%d: %d\n", i, p[i]);
75         order = make_array(number);
76
77         result = min_mult(number, p, order);
78         printf("\nOptimal chain Order: ");
79         print_order(1, number, order);
80         printf("\nMinimum number of computations: %d\n", result);
81         printf("\nfirst array\n");
82         for (i = 0; i < p[0]; i++)
83         {
84                 for (j = 0; j < p[1]; j++)
85                 {
86                         a[i][j] = rand()%10;
87                         printf("%d ", a[i][j]);
88                 }
89                 printf("\n");
90         }
91
```

- The main function outputs the number of matrices to be calculated first, and the row and column values (p) of each matrix.
- And it initializes the order array through a function called make array.
- The minimum number of computation value is calculated with a function called min_mult and an optical chain order is output through the print_order function.

Problem2 - code

```
92          printf("\nsecond array\n");
93          for (i = 0; i < p[1]; i++)
94          {
95                  for (j = 0; j < p[2]; j++)
96                  {
97                          b[i][j] = rand()%10;
98                          printf("%d ", b[i][j]);
99                  }
100                 printf("\n");
101         }
102         printf("\nthird array\n");
103         for (i = 0; i < p[2]; i++)
104         {
105                 for (j = 0; j < p[3]; j++)
106                 {
107                         c[i][j] = rand()%10;
108                         printf("%d ", c[i][j]);
109                 }
110                 printf("\n");
111         }
112         for(i = 0; i < p[1]; i++)
113         {
114                 for(j = 0; j < p[3]; j++){
115                         tmp[i][j] = 0;
116                         for(k = 0; k < p[2]; k++)
117                                 tmp[i][j] += b[i][k] * c[k][j];
118                 }
119         }
```

- The three matrices to be calculated were made from random number generation.
- First, calculate the second and third matrices, and then multiply the results with the first matrix.

Problem2 – code

```
120         for(i = 0; i < p[0]; i++)
121         {
122                 for(j = 0; j < p[3]; j++){
123                         final[i][j] = 0;
124                         for(k = 0; k < p[1]; k++)
125                                 final[i][j] += a[i][k] * tmp[k][j];
126                 }
127         }
128         printf("\nResult array\n");
129         for(i = 0; i < p[0]; i++){
130                 for(j = 0; j < p[3]; j++)
131                         printf("%d ", final[i][j]);
132                 printf("\n");
133         }
134         return 0;
135 }
```

Problem2 - code

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

int **make_array(int n)
{
        int i;
        int **arr = (int**)malloc(0);
        for (i = 0; i < n; i++)
                arr[i] = (int*)malloc(sizeof(int)*n);
        return arr;
}

void print_order(int i, int j, int *order[])
{
        int k;
        if(i == j)
        {
                printf("A%d", i);
        }
        else {
                k = order[i][j];
                printf("(");
                print_order(i,k,order);
                print_order(k+1, j,order);
                printf(")");
        }
}
```

Problem2 - code

```
int min_mult(int n, int p[], int *order[])
{
        int i, j, k, diag, min_k = 0;
        int **M, temp = 0;
        M = make_array(n+1);
        for(i = 1; i <= n; i++)
                M[i][i] = 0;
        for(diag = 1; diag <= n-1; diag++)
        {
                for(i = 1; i <= n - diag;i++) {
                        j = i + diag;
                        for(k = i; k <= j-1; k++) {
                                M[i][j] = M[i][k] + M[k+1][j] + p[i-1] * p[k] * p[j];
                                if(i == k) {
                                        temp = M[i][j];
                                        min_k = k;
                                }
                                else if(M[i][j] > temp) {
                                        M[i][j] = temp;
                                }
                                else
                                        min_k = k;
                        }
                        order[i][j] = min_k;
                }
        }
        return M[1][n];
}
```

Problem2 - result

```
[~/2022_1/Algorithm/assignment4]$ gcc problem2.c
[~/2022_1/Algorithm/assignment4]$ ./a.out
Number of array: 3
p0: 5
p1: 3
p2: 7
p3: 10

Optimal chain Order: (A1(A2A3))
Minimum number of computations: 360

first array
7 9 3
8 0 2
4 8 3
9 0 5
2 2 7

second array
3 7 9 0 2 3 9
9 7 0 3 9 8 6
5 7 6 2 7 0 3

third array
9 9 9 1 7 2 3 6 5 5
8 1 4 7 1 3 8 4 8 0
4 6 0 3 2 6 9 4 1 3
7 8 8 3 8 1 5 3 5 4
3 6 5 9 5 4 9 1 7 5
5 4 1 8 8 3 5 2 2 6
6 7 8 4 1 7 1 8 7 8

Result array
4241 3886 3530 3682 2828 2777 3944 2967 3775 3106
1908 1734 1416 1562 928 1524 1906 1534 1638 1356
3398 3121 2876 2975 2352 2150 3155 2317 3049 2482
2636 2410 2000 2178 1341 2039 2697 2053 2302 1828
2156 1999 1784 1857 1352 1436 2233 1485 1969 1450
```

Problem2 - code