MODUL 9
RECURSIVE FUNCTION

TUJUAN

1. Memahami konsep rekursif langsung dan tidak langsung,
2. Memahami aplikasi dari fungsi rekursif dalam memecahkan masalah pemrograman,
3. Memahami kelebihan dan kekurangan fungsi rekursif, serta
4. Memahami pemecahan kasus rekursif dan implementasinya di dalam program.

DASAR TEORI

Rekursi (bahasa Inggris: *recursion*) adalah suatu metode yang lazim digunakan di matematika dan pemrograman untuk menyederhanakan algoritma suatu proses. Di dalam pemrograman, fungsi rekursif merupakan fungsi yang mengandung fungsi itu sendiri, sehingga menimbulkan perulangan di dalam fungsi tersebut. Karena proses perulangan terjadi di dalam fungsi itu sendiri, kita tidak lagi menggunakan *for*, *while*, dan *do-while*.

Sebuah fungsi rekursif wajib memiliki definisi *base case* atau *termination point*. *Base case* adalah titik atau kondisi di mana fungsi tersebut akan berhenti. Tanpa adanya *base case*, maka akan terjadi *endless recursion*. Pada kasus terparah, *endless recursion* dapat memenuhi RAM PC (pada komputer tua), dan memaksa kita *force shutdown*/*restart* PC. Pada PC modern saat ini, *endless recursion* hanya akan mengakibatkan program crash.

EXAMPLE

Sebuah truk mengangkut sebuah truk yang kemudian mengangkut sebuah mobil. Ini merupakan sebuah fungsi rekursi dengan *base case* yang terdefinisi dengan benar.

Sebuah truk mengangkut sebuah truk yang mengangkut sebuah truk …. Ini merupakan sebuah fungsi rekursif dengan *base case* yang tidak terdefinisi dengan benar, sehingga menimbulkan *overflow*.

Info:
Untuk lebih memahami algoritma rekursif, silakan salin dan jalankan berbagai code di bawah ini, jangan lupa melakukan include library `stdio.h` dan `stdlib.h` di bagian atas pekerjaan. Tidak perlu membagi 3 file, cukup masukkan semua kode pada 1 file saja (contoh: main.c).

---

a. *Rekursif langsung dan tidak langsung*

Berdasarkan pemanggilnya, terdapat dua jenis fungsi rekursif yang umum digunakan, yakni:

- Fungsi rekursif langsung

  Fungsi rekursif langsung berarti di dalam fungsi tersebut, terjadi **pemanggilan terhadap dirinya sendiri**. Fungsi jenis ini merupakan fungsi rekursif yang paling sering digunakan.

CODE	OUTPUT

`void directRecursion(int n) {`
`  if (n > 0) {`
`  printf("%d ", n);`
`  directRecursion(n - 1);`
`  }`
`}`
 	
`int main() {`
`  int n = 5;`
`  directRecursion(n);`
`  return 0;`
`}`
	`5 4 3 2 1`

- Fungsi rekursif tidak langsung

  Fungsi rekursif tidak langsung berarti di dalam fungsi tersebut (sebut saja fungsi *A*), terjadi **pemanggilan terhadap fungsi lain** (sebut saja fungsi *B*), yang dapat saja memanggil fungsi lain juga (sebut saja fungsi *C*), namun pada akhirnya akan kembali memanggil fungsi *A*.

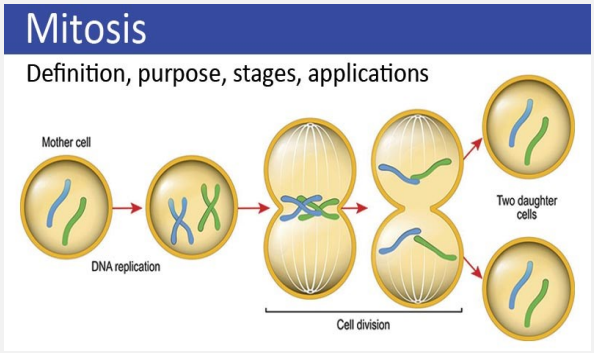
CODE	OUTPUT

`void indirectRecursionA(int n);`
`void indirectRecursionB(int n);`
 	
`void indirectRecursionA(int n) {`
`  if (n > 0) {`
`  printf("%d ", n);`
`  indirectRecursionB(n - 1);`
`  }`
`}`
	`20 19 9 8 4 3 1`

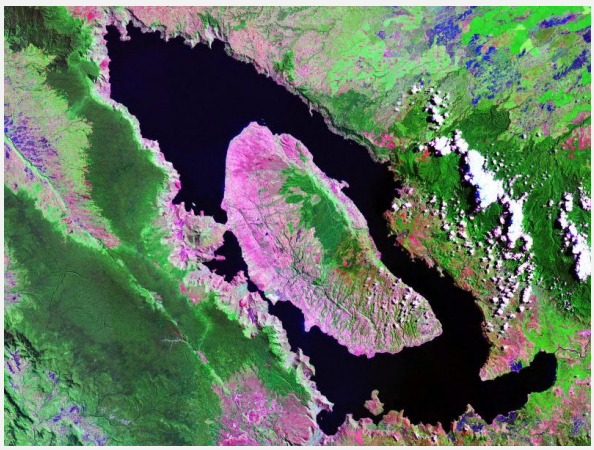
```c
void indirectRecursionB(int n) {
    if (n > 1) {
        printf("%d ", n);
        indirectRecursionA(n / 2);
    }
}

int main() {
    int n = 20;
    indirectRecursionA(n);
    return 0;
}
```

EXAMPLE



Mitosis merupakan proses pembelahan sel yang akan menghasilkan sel baru yang secara genetik sama dengan sel inangnya, sehingga merupakan contoh rekursif langsung.



Pulau Samosir merupakan sebuah pulau di Danau Toba, di Pulau Sumatra, merupakan contoh rekursi tidak langsung. *Recursive islands and lakes: en.wikipedia.org/wiki/Recursive_islands_and_lakes*

b. Aplikasi fungsi rekursif

Salah ssatu contoh kasus fungsi rekursif yang paling sering digunakan adalah pada perhitungan faktorial. Namun sebenarnya, berbagai jenis perulangan iterative dapat juga diekspresikan dalam bentuk rekursif.

1) Perhitungan faktorial (konsep rekursif dalam bentuk fungsi)

Perhatikan contoh perhitungan faktorial berikut:

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$, hasilnya adalah 120
- $4! = 4 \cdot 3 \cdot 2 \cdot 1$, hasilnya adalah 24
- $2! = 2 \cdot 1$, hasilnya adalah 2
- $1! = 1$, hasilnya adalah 1

Dari contoh tersebut, dapat terlihat suatu pola. Contohnya 5! dapat dirumuskan menjadi:

$$5! = 5 \cdot 4!$$

Terlihat pula bahwa 4! Adalah $4 \cdot 3!$, sehingga terdapat rumus umum, yakni:

$$n! = n \cdot (n-1)!$$

Rumus umum perhitungan faktorial didefinisikan sebagai berikut:

$$n! = \begin{cases} 1 & jika\ n = 0 \\ n \cdot (n-1)! & jika\ n > 0 \end{cases}$$

Dengan demikian, kita dapat membuat algoritmanya sebagai berikut:

Algoritma non-rekursif (iteratif):

CODE	OUTPUT

`int faktorialIterasi(int n) {`
`  int i, hasil = 1;`
`  for (i=1; i<=n; i++) {`
`  hasil = hasil * i;`
`  }`
`  return hasil;`
`}`
 	
`int main() {`
`  int n = 5;`
`  printf("Faktorial dari %d adalah %d", n, faktorialIterasi(n));`
`  return 0;`
`}`
	`Faktorial dari 5 adalah 120`

Algoritma rekursif:

Coba bandingkan: apakah algoritma non-rekursif atau algoritma rekursif lebih mudah dipahami dan lebih sesuai dengan rumus umum perhitungan faktorial.

CODE	OUTPUT

`int faktorialRekursif(int n) {`
`  if (n == 0) {`
`  return 1;`
`  } else {`
`  return n * faktorialRekursif(n - 1);`
`  }`
`}`
	`Faktorial dari 5 adalah 120`

```c
int main() {
    int n = 5;
    printf("Faktorial dari %d
adalah %d", n,
faktorialRekursif(n));
    return 0;
}
```

2) Mencetak (printf) sebuah daftar (konsep rekursif dalam bentuk prosedur)

Rekursif tidak hanya dapat dilakukan dengan fungsi, tetapi dapat juga dilakukan dalam bentuk prosedur.

Apabila kita memiliki sebuah daftar (berupa *array of strings*), kita dapat mencetaknya menggunakan fungsi rekursif.

Algoritma non-rekursif (iteratif)

CODE	OUTPUT

```c
#define MAX 5
typedef char string[100];

void printAllIterative(string
list[]) {
    int i;
    for (i = 0; i < MAX; i++) {
        printf("%s\n", list[i]);
    }
}

int main() {
    string list[MAX] = {
        "1. Tuangkan air",
        "2. Tuangkan susu",
        "3. Tambahkan 1 sdt kopi",
        "4. Aduk hingga merata",
        "5. Tuangkan air panas"
    };
    // Print semua item
    printAllIterative(list);
    return 0;
}
```

```
1. Tuangkan air
2. Tuangkan susu
3. Tambahkan 1 sdt kopi
4. Aduk hingga merata
5. Tuangkan air panas
```

Algoritma rekursif

CODE	OUTPUT

`#define MAX 5`
`typedef char string[100];`
 	
`void printAllRecursive(string list[], int index) {`
`  printf("%s\n", list[index]);`
 	
`  if (index == MAX - 1) {`
`  // Base case: do nothing`
`  } else {`
`  // Lanjutkan, index + 1`
`  printAllRecursive(list, index + 1);`
`  }`
`}`
 	
`int main() {`
`  string list[MAX] = {`
`  "1. Tuangkan air",`
`  "2. Tuangkan susu",`
`  "3. Tambahkan 1 sdt kopi",`
`  "4. Aduk hingga merata",`
`  "5. Tuangkan air panas"`
`  };`
`  // Print semua item`
`  printAllRecursive(list, 0);`
`  return 0;`
`}`
	`1. Tuangkan air`
	`2. Tuangkan susu`
	`3. Tambahkan 1 sdt kopi`
	`4. Aduk hingga merata`
	`5. Tuangkan air panas`

Apabila diamati, terdapat beberapa perbedaan pada algoritma rekursif dan non-rekursif.

Kriteria	Iteratif	Rekursif
Parameter prosedur	`void printAllIterative(string list[])` Hanya ada 1 parameter: list yang ingin dicetak.	`void printAllRecursive(string list[], int index)` Ada 2 parameter: list yang ingin dicetak dan indeks saat ini.

Body prosedur	`int i;`
	`for (i = 0; i < MAX; i++) {`
	`  printf("%s\n", list[i]);`
	`}`
	Kita mendeklarasikan variabel `i`, sebagai penanda indeks, kemudian
		`printf("%s\n", list[index]);`
		
		`if (index == MAX - 1) {`
		`  // Base case: do nothing`
		`} else {`
		`  // Lanjutkan, index + 1`
		`  printAllRecursive(list, index + 1);`
		`}`

	menggunakan *for loop* untuk mencetak item di list.	Kita terlebih dahulu mencetak item di list, kemudian memeriksa apakah kita sudah mencapai ujung list (*base case*). Apabila belum mencapai ujung list, lanjutkan algoritma rekursif.
Pemanggi-lan prosedur	`printAllIterative(list);` Sesuai dengan parameter yang ada, kita hanya perlu menyertakan list yang ingin dicetak.	`printAllRecursive(list, 0);` Karena ada parameter `index`, maka perlu juga kita sertakan indeks pertama dari list tersebut, yakni indeks 0.

CHALLENGE	EXPECTED OUTPUT

Coba balikkan urutan cetak dari daftar di atas menggunakan algoritma rekursif.
Hasilnya harus sama dengan di samping →
	`5. Tuangkan air panas`
	`4. Aduk hingga merata`
	`3. Tambahkan 1 sdt kopi`
	`2. Tuangkan susu`
	`1. Tuangkan air`

c. *Kelebihan dan kekurangan fungsi rekursif*

Fungsi rekursif tidak dapat digunakan untuk menyelesaikan semua masalah. Ada beberapa kelebihan dan kekurangan dari fungsi rekursif:

- Kelebihan
  1) Lebih sederhana dibandingkan perulangan iteratif
  2) Mengurangi pemanggilan fungsi berlebihan
  3) Dalam beberapa kasus, lebih mudah digunakan untuk menyelesaikan suatu masalah (menghitung pangkat, faktorial, Fibonacci, deret, dll).
- Kekurangan
  1) Membutuhkan memori yang lebih besar
  2) Kasus terburuk jika *endless recursion*: PC *crash*

Modul ini (fungsi rekursif) merupakan konsep dasar dari modul-modul selanjutnya, yakni Linked List Recursive, Binary Tree, Searching, dan Sorting. Pemahaman akan konsep algoritma rekursif penting bagi modul-modul ke depannya.

Bacaan lanjutan:

- https://www.geeksforgeeks.org/introduction-to-recursion-data-structure-and-algorithm-tutorials/
- https://www.geeksforgeeks.org/recursive-functions/

Sumber dan referensi:

- https://en.wikipedia.org/wiki/Recursion_(computer_science)
- https://www.programiz.com/cpp-programming/recursion
- https://www.reddit.com/r/ProgrammerHumor/search/?q=recursion&source=recent&restrict_sr=1&include_over_18=1&sort=top&t=all

To understand recursion,
one must first understand recursion.

STEPHEN HAWKING

GUIDED

header.h

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

#define MAX 5
typedef char string[100];

typedef struct {
    string nama;
    string npm;
    float ipk;
} Mahasiswa;

void initData(Mahasiswa M[], int index);
int getEmptyIndex(Mahasiswa M[], int index);
void printAll(Mahasiswa M[], int index);
int isEmpty(Mahasiswa M[], int index);
```

source.c

```c
#include "header.h"

// Inisialisasi array mahasiswa
void initData(Mahasiswa M[], int index) {
    strcpy(M[index].nama, "");
    strcpy(M[index].npm, "");
    M[index].ipk = 0.0;

    // Lanjutkan fungsi rekursif
    if(index >= MAX - 1) {
        // do nothing
    } else {
        initData(M, index + 1);
    }
}

// Mendapatkan index yang masih kosong
int getEmptyIndex(Mahasiswa M[], int index) {
    if (strcmpi(M[index].npm, "") == 0) {
        return index;
    }

    // Lanjutkan fungsi rekursif
    if(index >= MAX - 1) {
        return -1;
    } else {
        return getEmptyIndex(M, index + 1);
    }
}

// Mencetak semua data mahasiswa
void printAll(Mahasiswa M[], int index) {
    if (strcmpi(M[index].npm, "") != 0) {
        printf("\n\tMahasiswa ke-%d", index + 1);
        printf("\n\tNama\t: %s", M[index].nama);
        printf("\n\tNPM\t: %s", M[index].npm);
        printf("\n\tIPK\t: %.2f\n", M[index].ipk);
    }

... dilanjutkan
```

```
... dilanjutkan
39
40      // Lanjutkan fungsi rekursif
41      if (index >= MAX - 1) {
42          // do nothing
43      } else {
44          printAll(M, index + 1);
45      }
46  }
47
48  // Mengecek apakah array mahasiswa masih kosong
49  int isEmpty(Mahasiswa M[], int index) {
50      if (strcmpi(M[index].npm, "") != 0) {
51          return 0;
52      }
53
54      // Lanjutkan fungsi rekursif
55      if (index >= MAX - 1) {
56          return 1;
57      } else {
58          return isEmpty(M, index + 1);
59      }
60  }
```

main.c

```
1   #include "header.h"
2
3   int main(int argc, char *argv[]) {
4       Mahasiswa M[MAX], temp;
5       int menu, index;
6
7       // inisialisasi array Mahasiswa, mulai dari indeks 0
8       initData(M, 0);
9
10      do {
11          system("cls");
12          printf("--- Guided: Mahasiswa ---");
13          printf("\n[1] Input Mahasiswa");
14          printf("\n[2] Tampil Mahasiswa");
15          printf("\n[0] Keluar");
16          printf("\n>>> "); scanf("%d", &menu);
17          switch(menu) {
18              case 1:
19                  printf("\n\t[1] Input Mahasiswa\n");
20
21                  // Cari index yang masih kosong
22                  index = getEmptyIndex(M, 0);
23                  if (index != -1) {
24                      // Masih ada index yang kosong, masukkan data
25                      while(1) {
26                          printf("\tNPM   : "); fflush(stdin); gets(temp.npm);
27
28                          // Error handling
29                          if (strlen(temp.npm) == 0) {
30                              // NPM tidak boleh kosong
31                              printf("\t[-] NPM tidak boleh kosong!\n");
32                          } else {
33                              // Tidak ada lagi error, berarti bisa lanjut
34                              break;
35                          }
36                      }
37                      printf("\tNama  : "); fflush(stdin); gets(temp.nama);
38                      printf("\tIPK   : "); scanf("%f", &temp.ipk);
39
... dilanjutkan
```

```
... dilanjutkan
39
40                      // Simpan data mahasiswa kembali ke array
41                      M[index] = temp;
42                      printf("\n\t\t[+] Data berhasil disimpan!");
43                  } else {
44                      // Apabila index == -1, berarti tidak ada lagi slot yang kosong
45                      printf("\n\t[-] Data sudah penuh!");
46                  }
47                  break;
48
49              case 2:
50                  printf("\n\t[2] Tampil Mahasiswa\n");
51
52                  if (!isEmpty(M, 0)) {
53                      // Cetak semua data mahasiswa dari indeks 0
54                      printAll(M, 0);
55                  } else {
56                      printf("\n\t[-] Data masih kosong!");
57                  }
58                  break;
59
60              case 0:
61                  // Keluar
62                  printf("\n\t[+] Terima kasih!");
63                  printf("\n\t<nama> - <NPM> - Praktikum ISD <kelas>");
64                  break;
65              default:
66                  printf("\n\t[-] Menu tidak tersedia");
67                  break;
68          }
69          getch();
70      } while(menu != 0);
71      return 0;
72  }
```

Hint dan Tips UGD:

1. Tidak diperkenankan menggunakan perulangan iterasi (*for…, while…, do…while*) dalam pembuatan berbagai prosedur dan fungsi **kecuali** untuk menu (di main.c) atau error handling (contoh di main.c line 25-36).
2. Pahamilah cara kerja algoritma rekursif dengan baik, karena akan ada fungsi/prosedur *custom* (yang belum ada penerapannya di guided) yang perlu dibuat nanti.

Format Pengumpulan:

GD9_X_YYYYY

X = kelas

Y = 5 digit terakhir NPM