COMPUTER ENGINEERING

Laboratory
Multimedia and Internet of Things
Computer Engineering Department
Institut Teknologi Sepuluh Nopember

Basic Programming Practicum

Pointers, Structs, and Algorithms

2023

1 Goals

- Students are able to understand the concept of pointers in C.
- Students are able to create and call a struct in C.
- Students are able to understand about sorting algorithm in C.
- Students are able to understand about searching algorithm in C.
- Students are able to apply the conceptof searching and sorting algorithm in C.

2 Pointers

2.1 Memory Address

Every variables, functions, structs, or any object in a program have their own memory allocation. Said Allocations are saved in certain memory addresses

If there are any variable `var` in your program, `&var` will return its address in memory

```
1     int var = 5;
2     printf("%d\n", var);
3     printf("%p\n", &var);
```

Catatan: Output may differ in each execution.

2.2 Introduction to Pinters

Pointers is a special variable that serves to save addresses, not value.

To declare a pointer variable, use $*$ operator between the data type and it's variable name.

```
1   #include <stdio.h>
2 int main()
3 {
4   int* p; // atau
5     int * p2;
6   return 0;
7 }
```

2.3 How Pointers Work

The following is to show how pointers work.

Listing 1: Program Example

```
1     #include <stdio.h>
2     int main()
3     {
4        int* pc, c;
5
6        c = 22;
7        printf("Address of c: %p\n", &c);
8        printf("Value of c: %d\n\n", c);  // 22
9
```

```c
    pc = &c;
    printf("Address of pointer pc: %p\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc); // 22

    c = 11;
    printf("Address of pointer pc: %p\n", pc);
    printf("Content of pointer pc: %d\n\n", *pc); // 11

    *pc = 2;
    printf("Address of c: %p\n", &c);
    printf("Value of c: %d\n\n", c); // 2
    return 0;
}
```

Explanation:

1. `int* pc, c;`

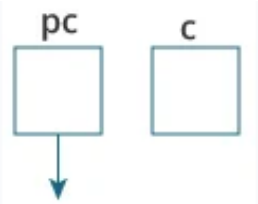

Figure 1

2. `c = 22;`

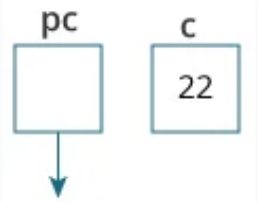

Figure 2

3. `pc = &c;`

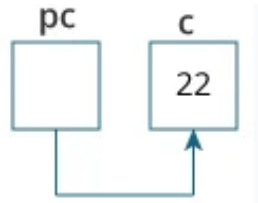

Figure 3

4. `c = 11;`

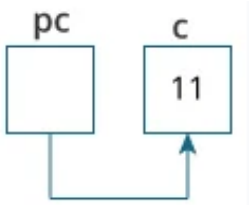

Figure 4

5. `*pc = 2;`

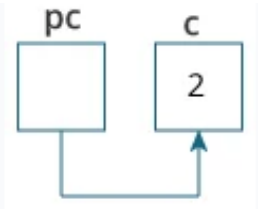

Figure 5

2.4 Double Pointer

A pointer could also point to another pointer variable. This is called a double pointer (pointer to pointer). To declare a double pointer, we use two $*$ operator between the data type and its variable name. The general usage of this double pointer variable is ti create a two dimensional array dinamically.

```
1     int **dbPtr;
```

Variabel dbPtr di atas menyimpan alamat memori dari variabel pointer lainnya.
Berikut contohnya

The variable dbPtr above contains a memory address of another pointer variable.
Take a look at the example below.

Listing 2: Double Pointer Example

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int var = 23;
6      int *ptr = &var;
7      int **dbPtr = &ptr;
8
9      printf("%d\n", **dbPtr);
10
11     return 0;
12 }
```

2.5 Pre-lab Assignment

1. Explain how to declare a pointer to a multidimensional array?

2. Create a program in C or C++ that implements void printMatrix(int **matrix, int rows, int cols) function to print out a 2D matrix using pointer to pointer. And then in the main function, create a 2D matrix and call the printMatrix function to print said matrix.

3 Struct

In C, struct is a collection of variables (could be in various types) under one name. Unlike an array that could only store elements with the same data type, struct is able to group elements with different data types.

3.1 Deklarasi Struct

Similar to variables, a struct must be declare first before use. Below is an example on how to declare a struct.

```
struct <struct_name> {
    <member_dataType> <member_name>;
    <member_dataType> <member_name>;
    <member_dataType> <member_name>;
    .
    .
    .
};

```

Berikut adalah contoh deklarasi struct berdasarkan kasus Mahasiswa. Below is a case example of struct declaration about students data.

```
struct Students
{
    char *name;
    char *address;
    int age;
};

```

Note: Use pointer * for string data type

After declaration, a struct will be its own data type. In this case, **Students** data type will be its own new data type with members such as `name`, `address`, dan `age`. Take a look at an example below to declare a variable with the struct data type.

```
struct <struct_name> <variable_name>;
```

Example:

```
struct Students mhs1;
struct Students mhs2;
```

Contoh di atas menunjukkan terdapat dua variabel `mhs1` dan `mhs2` bertipe struct `Mahasiswa`. The example above shows that there are two variables `mhs1` and `mhs2 with \verb`Mahasiswa| struct data type.

3.2 Struct Member Access

Bagaimana cara untuk mengakses member dari variabel struct yang telah dibuat?
Untuk mengakses member-member dari struct, digunakan operator dot (.) setelah nama variabelnya.
How do you access members from a struct variable?
To access members from a struct, we use a dot (.) operator after its variable name.

```
<variable_name>.<member_name>
```

Example:

```
mhs1.age = 20;
mhs1.nama = iqbal;

mhs2.nama = fatur;
mhs2.age = 21;
```

3.3 Pre-lab Assignment

1. Create a struct that represent informations about a students, the informations their name, student ID, and GPA. And then, create a program in insert student informations, display them, and calculate the average GPAs of a certain number of students

2. Anda diberikan struct yang merepresentasikan titik dalam sistem koordinat dua dimensi (x, y). Buatlah sebuah program C untuk menghitung jarak antara dua titik yang diinputkan oleh pengguna menggunakan rumus jarak Euclidean.

3. Create a struct that represents coordinate points in a two dimensional plane (x,y). And then create a C program to calculate the distance between two points that a user inputs with a Euclidean formula.

4 Sorting algorithm

Sorting is a process of arranging or organizing data.
There are two types of data sorting, namely :

1. Ascending (small to large).

2. Descending (large to small).

4.1 Bubble Sort

Bubble sort is a sorting algorithm that compares two adjacent data points and swaps them until they are in the desired order. Bubble sort utilizes iteration method. iteration is a process of repeating a loop as many times as there is known data. Essentially, during each iteration, a comparison is made between two data points.

Figure 6

Listing 3: Bubble Sort Implementation

```
void swap ( int * xp , int * yp ) {
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}

void bubbleSort(int arr[], int n) {
   int i, j, swapped;        // optimized with bool 'swapped':
   for (i = 0; i < n-1; i++) {
      swapped = 0;
      for (j = 0; j < n-i-1; j++) {
         if (arr[j] > arr[j+1]) {
            swap(&arr[j], &arr[j+1]);
            swapped = 1;
         }
      }
      if (swapped == 0)
         break;
   }
}
```

4.2 Insertion Sort

Insertion sort is a sorting technique that involves repeatedly inserting or placing each element. Imagine reshuffling a deck of cards to organize it.

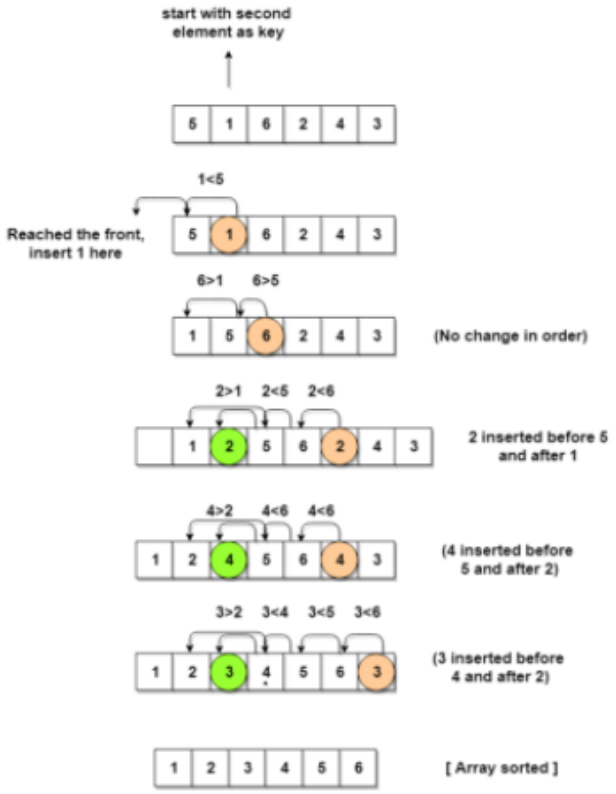

Figure 7

Listing 4: Insertion Sort Implementation

```
void insertionSort(int arr[]. int n) {
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i-1;

      while (j >= 0 && arr[j] > key) {
         arr[j+1] = arr[j];
         j = j-1;
      }
      arr[j+1] = key;
   }
}
```

Note: There are other sorting algorithms. Look it up individually

4.3 Pre-lab Assignment

1. Sort the following array using Bubble Sort:

   Array: [5, 2, 9, 1, 5, 6]

2. Hitung kompleksitas waktu (Big O) dari algoritma Insertion Sort saat mengurutkan sebuah array dengan panjang n, dan jelaskan bagaimana kompleksitas ini dihitung.

3. Calculate the time complexity (Big O) of the Insertion Sort algorithm when soring an array of length n, and explain how this complexity is calculated

5 Searching Algorithm

Searching is a process of looking the desired data.

5.1 Linear Search

Linear Search bekerja dengan melakukan pengecekan kepada semua elemen yang ada.
Secara garis besar, cara kerja Linear Search adalah:

Linear Search works by checking all of the existing elements.
Essentially, the working principle of Linear Search is:

1. Checking items one by one.
2. When it is found, the program will execute any statements that needs a condition of an item to be found.
3. If the algorithm had checked every single data, then the desired item does not exist.

Listing 5: Linear Search Implementation

```
14 int linearSearch(int arr[], int n, int item) {
15     int i;
16     for(i = 0; i < n; ++i) {
17         if(item == arr[i])
18           return 1;
19     }
20     return -1;
21 }
```

5.2 Binary Search

Binary Search is a searching technique where in each iteration we separate the searching space to half the initial searching space until we find the desired item.

Listing 6: Binary Search Implementation

```
22 bool f(int k, int a, int b, int n) {
23    return ((k/a) * (b/a) >= n);
24 }
25
26 int binser(int a, int b, int n) {
27    int l = 1;
28    int r = 100000;
29    while (r - l > 1) {
30       int mid = (l + r) >> 1;
31       bool can = f(mid);
32       if(can)
33          r = mid;
34       else
35          l = mid + 1;
36    }
37    if (can(l))
```

```
38        return l;
39    else
40        return r;
41 }
```

Note: There are other sorting algorithms. Look it up individually

5.3 Pre-lab Assignment

1. Given the following list of names: ["Alice", "Bob", "Charlie", "David", "Eve", "Frank"]. Use the Binary Search algorithm to look for the name "Eve" in this list. If found, how many steps are required?

2. Explain the difference between Linear Search (Sequential Search) and Binary Search. in what scenario that you choose one method over the other?