COMPUTER ENGINEERING

Laboratorium
Multimedia dan Internet of Things
Departemen Teknik Komputer
Institut Teknologi Sepuluh Nopember

Praktikum
Dasar Pemrograman

Pointer, Struct, dan Algoritma

2023

1 Tujuan

- Mahasiswa mengerti tentang konsep pointer pada bahasa pemrograman C.
- Mahasiswa mengerti cara membuat dan memanggil struct pada bahasa pemrograman C.
- Mahasiswa mengerti tentang algoritma sorting pada bahasa pemrograman C.
- Mahasiswa mengerti tentang algoritma searching pada bahasa pemrograman C.
- Mahasiswa mampu mengaplikasikan konsep algoritma searching dan sorting pada bahasa pemrograman C.

2 Pointer

2.1 Alamat Memori

Setiap variabel, fungsi, struct, ataupun objek lain yang dibuat dalam program mempunyai lokasi masing-masing pada memori. Alokasi setiap variabel disimpan dalam alamat memori tertentu.

Jika terdapat variabel `var` di program Anda, `&var` akan memberi alamatnya di memori.

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

Catatan: Output bisa berbeda-beda di tiap eksekusi.

2.2 Pengenalan Pointer

Pointer (variabel penunjuk) adalah variabel khusus yang digunakan untuk menyimpan alamat, bukan nilai.

Deklarasi variabel pointer menggunakan operator $*$ di antara tipe data dan nama variabelnya.

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

2.3 Cara Kerja Pointer

Berikut adalah cara kerja dari pointer.

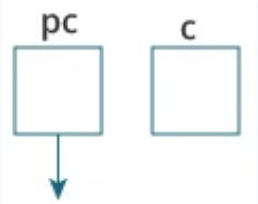
Listing 1: Contoh Program Pointer

```
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
10      pc = &c;
11      printf("Address of pointer pc: %p\n", pc);
12      printf("Content of pointer pc: %d\n\n", *pc); // 22
13
14      c = 11;
15      printf("Address of pointer pc: %p\n", pc);
16      printf("Content of pointer pc: %d\n\n", *pc); // 11
17
18      *pc = 2;
19      printf("Address of c: %p\n", &c);
20      printf("Value of c: %d\n\n", c); // 2
21      return 0;
22  }
```

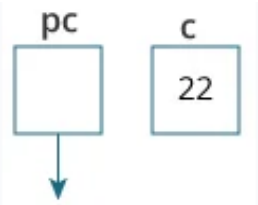
Penjelasan Program:

1. `int* pc, c;`



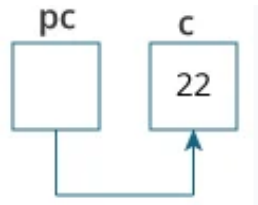
Gambar 1

2. `c = 22;`



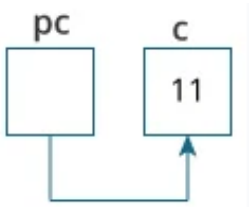
Gambar 2

3. `pc = &c;`



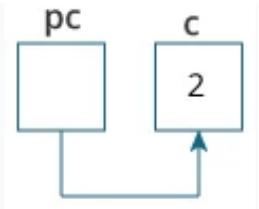
Gambar 3

4. `c = 11;`



Gambar 4

5. `*pc = 2;`



Gambar 5

2.4 Double Pointer

Variabel pointer juga dapat menunjuk variabel pointer lainnya. Hal ini disebut dengan double pointer (pointer to pointer). Untuk mendeklarasikan variabel double pointer, digunakan dua simbol *. Kegunaan paling umum dari variabel double pointer adalah untuk membuat array dua dimensi secara dinamis.

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

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

Listing 2: Contoh Double Pointer

```
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 Tugas Pendahuluan

1. Bagaimana cara mendeklarasikan pointer ke array multidimensi?

2. Buatlah program dalam bahasa C atau C++ yang mengimplementasikan fungsi void printMatrix(int **matrix, int rows, int cols) untuk mencetak matriks 2D menggunakan pointer ke pointer.

Lalu, dalam fungsi main, buatlah matriks 2D dan panggil fungsi printMatrix untuk mencetak matriks tersebut.

3 Struct

Dalam pemrograman C, struct (atau struktur) adalah kumpulan variabel (bisa dari tipe berbeda) di bawah satu nama. Tidak seperti array yang hanya dapat menyimpan elemen dengan tipe data sama, struct dapat mengelompokkan elemen dengan tipe data yang berbeda-beda.

3.1 Deklarasi Struct

Seperti variabel, struct harus dideklarasikan terlebih dahulu sebelum bisa digunakan. Pendeklarasian struct menggunakan sintaks sebagai berikut.

```
struct <nama_struct> {
    <tipe_data_member> <nama_member>;
    <tipe_data_member> <nama_member>;
    <tipe_data_member> <nama_member>;
    .
    .
    .
};
```

Berikut adalah contoh deklarasi struct berdasarkan kasus Mahasiswa.

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

Catatan: Menggunakan pointer * untuk data string

Setelah dideklarasikan, sebuah struct akan menjadi tipe data baru. Maka dalam kasus ini, struct Mahasiswa di sini menjadi tipe data baru dengan member-member berupa `nama`, `address`, dan `age`. Untuk membuat variabel dengan tipe data struct, dilakukan dengan sintaks berikut.

```
    struct <nama_struct> <nama_variabel>;
```

Contoh:

```
    struct Mahasiswa mhs1;
    struct Mahasiswa mhs2;
```

Contoh di atas menunjukkan terdapat dua variabel `mhs1` dan `mhs2` bertipe struct `Mahasiswa`.

3.2 Akses Member Struct

Bagaimana cara untuk mengakses member dari variabel struct yang telah dibuat?
Untuk mengakses member-member dari struct, digunakan operator dot (.) setelah nama variabelnya.

```
<nama_variabel>.<member_struct>
```

Contoh:

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

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

3.3 Tugas Pendahuluan

1. Buatlah sebuah struct yang merepresentasikan informasi tentang seorang mahasiswa, yang memiliki nama, nim, dan nilai IPK. Kemudian, buatlah program untuk menginput data mahasiswa, menampilkan data mahasiswa, dan menghitung rata-rata IPK dari sejumlah mahasiswa.

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.

4 Algoritma Sorting

Sorting merupakan suatu proses penyortiran atau pengurutan sebuah data.
Terdapat 2 macam pengurutan data pada sorting yaitu :

1. Berdasarkan ascending (kecil ke besar).

2. Berdasarkan Descending (besar ke kecil).

4.1 Bubble Sort

Bubble sort merupakan algoritma pengurutan yang membandingkan dua data yang berdekatan dan menukarnya sampai tidak dalam urutan yang diinginkan. Bubble sort menggunakan teknik iterasi. Iterasi merupakan proses melakukan perulangan sebanyak data yang diketahui. Intinya pada iterasi melakukan perbandingan antara dua data.

Gambar 6

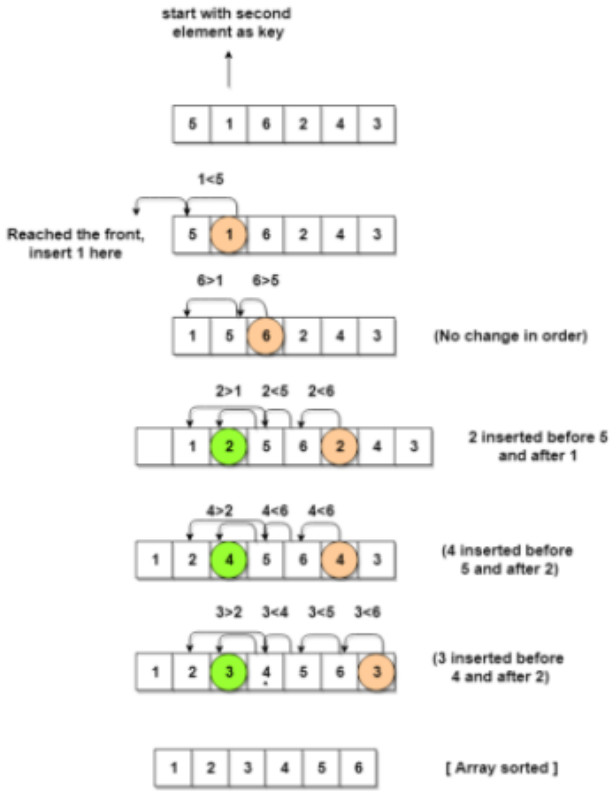
Listing 3: Implementasi Bubble Sort

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

void bubbleSort(int arr[], int n) {
   int i, j, swapped;        // dioptimasi dengan 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 merupakan teknik sorting dengan cara menyisipkan atau memasukan setiap elemen secara berulang berulang. Konsep insertion sort bisa diibaratkan sebuah kartu.



Gambar 7

Listing 4: Implementasi Insertion Sort

```
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;
   }
}
```

Catatan: Terdapat berbagai algoritma sorting lain. Pelajari secara mandiri

4.3 Tugas Pendahuluan

1. Urutkan array berikut menggunakan algoritma 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.

5 Algoritma Searching

Searching merupakan proses pencarian sebuah data yang diinginkan.

5.1 Linear Search

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

1. Memeriksa item satu per satu.
2. Apabila ditemukan, maka ketemu.
3. Jika sampai akhir belum ditemukan, maka item yang dicari tidak ada.

Listing 5: Implementasi Linear Search

```
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 adalah teknik pencarian di mana untuk setiap iterasinya kita membagi space pencarian menjadi hanya setengah dari space pencarian awal hingga kita menemukan yang kita cari.

Listing 6: Implementasi Binary Search

```
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 }
```

Catatan: Terdapat berbagai algoritma searching lain. Pelajari secara mandiri

5.3 Tugas Pendahuluan

1. Anda memiliki daftar nama berikut: ["Alice", "Bob", "Charlie", "David", "Eve", "Frank"]. Gunakan algoritma binary search untuk mencari apakah nama "Eve" ada dalam daftar ini. Jika ya, berapa langkah yang dibutuhkan?

2. Jelaskan perbedaan antara pencarian linear (sequential search) dan pencarian biner (binary search). Kapan Anda akan memilih salah satu metode ini daripada yang lain?