



Struktur Data
Pendidikan Teknik Informatika
Universitas Negeri Padang
TIM DOSEN ©2022

JOBSHEET (JS-02)

Struct dan Pointer

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa mampu mengaplikasikan konsep struct dalam menyelesaikan kasus.
2. Mahasiswa mampu mengaplikasikan konsep pointer dalam menyelesaikan kasus.

HARDWARE & SOFTWARE

1. Personal Computer
2. IDE: Dev C++

TEORI SINGKAT

A. Struct

Untuk menyimpan banyak data pada satu variable, biasanya kita menggunakan array. Array mampu menyimpan banyak data dalam satu variable, yaitu dengan ditempatkan dalam index yang berbeda. Tapi kita juga bisa menyimpan banyak data dalam satu index, bahkan dengan tipe data yang berbeda. Untuk dapat melakukan hal ini, kita menggunakan structure.

Structure adalah kumpulan dari beberapa variable yang dikelompokan dalam satu nama untuk kemudahan dalam penggunaan variable-variable tersebut. Structure mampu mengelompokan variable dengan tipe data yang berbeda dalam satu nama. Structure sangat diperlukan terutama dalam program-program yang besar. Karena, biasanya program-program yang besar menggunakan data yang kompleks, yang harus

dolah dan digunakan secara bersamaan. Strucutre pada beberapa bahasa pemrograman lain disebut "record", misalnya pascal.

Contoh penggunaan structure misalnya pada seorang mahasiswa. Dalam program, kita bisa membuat mahasiswa sebagai sebuah variable. tapi dalam dunia nyata mahasiswa memiliki banyak data, antara lain alamat, nama, umur, NIM, dll. Pada program yang besar, data-data yang dimiliki mahasiswa harus dikelola secara bersamaan. Dan pada program, kita akan sangat kerepotan apabila menggunakan satu buah variable saja. Untuk mengatasi masalah ini, kita memerlukan structure untuk mengelompokan data dan mempermudah dalam mengolahnya.

Operasi yang bisa dilakukan terhadap structure antara lain, penggandaan (copy), sebagai parameter fungsi, dan sebagai nilai kembalian dari fungsi. Structure juga bisa digabungkan dengan array, yaitu dengan memberi index pada nama structure.

Format deklarasi structure adalah sebagai berikut:

```
struct <NamaStructure>
{
    member
    member
    member
}
```

Struct merupakan suatu struktur data yang menggabungkan beberapa data dengan berbagai tipe data yang memiliki ukuran yang berbeda (heterogen) di kelompokan dalam satu deklarasi unik dan saling berkaitan, dengan format sbb :

```
struct model_name
{
    type1 element1;
    type2 element2;
    type3 element3;
    .
    .
} object_name;
```

Dimana model_name adalah nama untuk model tipe stukturnya dan parameter optional object_name merupakan identifier yang valid untuk objek struktur. Diantara kurung kurawal { } berupa tipe dan sub-identifier yang mengacu ke

elemen pembentuk struktur. Jika pendefinisian struktur menyertakan parameter model_name (optional), maka parameter tersebut akan menjadi nama tipe yang valid ekuivalen dengan struktur. Contoh :

```
struct products {  
    char name [30];  
    float price;  
};  
products apple;  
products orange, melon;
```

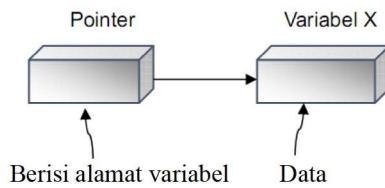
Didefinisikan model struktur products dengan dua field : name dan price, dengan tipe yang berbeda. Kemudian tipe struktur tadi (products) digunakan untuk mendeklarasikan tiga objek : apple, orange dan melon. Ketika dideklarasikan, products menjadi nama tipe yang valid seperti tipe dasar int, char atau short dan dapat mendeklarasikan objects (variables) dari tipe tersebut. Optional field yaitu object_name dapat dituliskan pada akhir deklarasi struktur untuk secara langsung mendeklarasikan object dari tipe struktur. Contoh :

```
struct products {  
    char name [30];  
    float price;  
} apple, orange, melon;
```

Sangat penting untuk membedakan antara structure model, dan structure object. model adalah type, dan object adalah variable. Kita dapat membuat banyak objects (variables) dari satu model (type). Struct dapat dideklarasikan secara bertingkat, yaitu salah satu field struct bertipe struct lainnya (nested Structure) . Selain itu struct juga dapat digabungkan dengan array, struct yang field-nya berupa array atau array yang setiap elemennya berupa structure.

B. Pointer

Pointer adalah tipe data yang digunakan untuk menunjuk ke suatu data. Suatu variabel yang bertipe pointer (variabel pointer) tidak berisi data, melainkan berisi alamat suatu data. Di dalam komputer setiap lokasi data mempunyai alamat yang khas. Gambar berikut contoh suatu pointer yang menunjuk ke suatu data.



Berdasarkan kondisi di atas, dimungkinkan untuk mengakses data pada variabel X melalui Pointer. Pointer biasa digunakan sehubungan dengan pembentukan variabel dinamis. Variabel Dinamis adalah variabel yang bisa dialokasikan di dalam memori atau dihapus dari memori ketika program dieksekusi. Hal seperti ini banyak dipakai pada struktur data seperti linked list dan pohon biner.

1. Mendeklarasikan Variabel Pointer

Suatu variabel pointer dideklarasikan dengan bentuk sebagai berikut:

tipedata *nama_variabel

dengan tipe data dapat berupa sembarang tipe data yang sudah dibahas sebelumnya. Adapun **nama_variabel** adalah nama dari variabel pointer. Sebagai contoh:

int *px;
char *pch1, *pch2;

Contoh pertama menyatakan bahwa px adalah variabel pointer yang menunjukkan ke suatu data bertipe int, sedangkan contoh kedua masing-masing pch1 dan pch2 adalah variabel pointer yang menunjukkan ke data bertipe char.

char *pch1, *pch2;
 menyatakan variabel pointer
tanda akhir pernyataan deklarasi
nama variabel pointer
tipe data yang ditunjuk oleh variabel pointer

2. Mengatur Pointer agar Menunjuk ke Variabel Lain (Address Operator -- &)

Ketika kita mendeklarasikan suatu variabel, kita tidak menentukan dimana variabel tersebut akan ditempatkan dimemori. Hal tersebut akan dilakukan oleh compiler dan sistem operasi pada saat runtime.

Kadang-kadang kita ingin mengetahui alamat dimana variabel kita ditempatkan, hal tersebut dapat dilakukan dengan mengawali identifier variabel tersebut dengan suatu *ampersand sign* (&), dimana dibaca sebagai "*address of*". contoh:

```
ted = &andy;
```

akan mengisi variabel **ted** dengan alamat dari variabel **andy**.

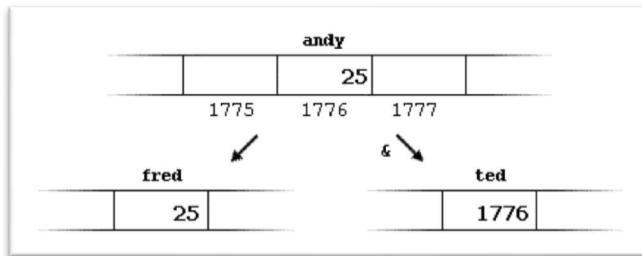
Misalnya variabel **andy** ditempatkan pada alamat memori **1776** dan penulisan berikut:

```
andy = 25;
```

```
fred = andy;
```

```
ted = &andy;
```

akan menghasilkan seperti diagram berikut :

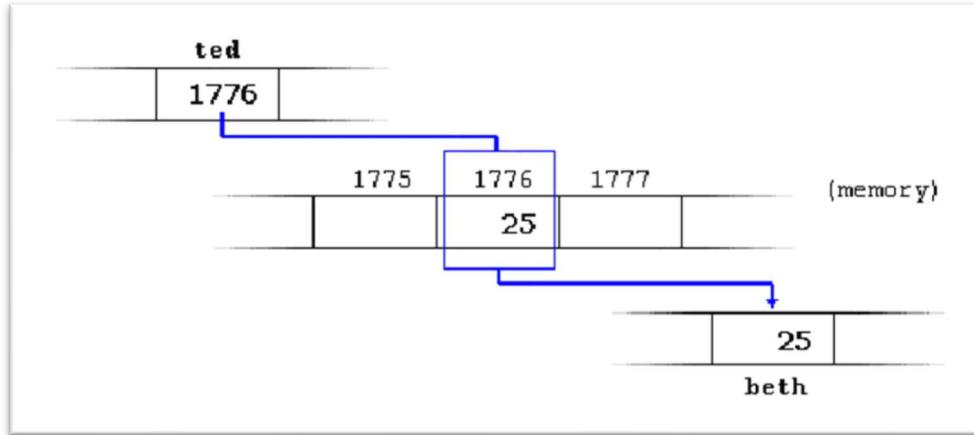


3. Mengakses Isi Suatu Variabel Melalui Pointer (Reference Operator -- *)

Dengan menggunakan suatu pointer kita dapat langsung mengakses nilai dari alamat yang tersimpan pada variabel dengan mengawalinya dengan operator asterisk (*), yang dibaca sebagai "value pointed by". lanjutan dari contoh sebelumnya:

```
beth = *ted;
```

(dimana dapat dibaca sebagai: "beth sama dengan value pointed by ted") **beth** akan berisi nilai **25**, karena **ted** adalah **1776**, dan nilai yang ditunjuk oleh **1776** adalah **25**.



4. Pointer dan Array

Dalam pemrograman C, definisi larik dituliskan: `type_name array_name [number_of_array]`, misal larik A bertipe integer dengan 10 anggota didefinisikan dengan `int A[10]`. Apa maksudnya? Dengan penulisan itu, maka diperintahkan kepada kompiler untuk menyediakan alamat memori sebesar $10 * \text{sizeof(int)}$. Bila ukuran int adalah 4 byte, maka compiler akan mengalokasikan sebesar $10 * 4$ byte = 40 byte memori untuk A. Maka penggambarannya dapat diilustrasikan sebagai berikut.

Array	Alamat Memori
A[0]	0xDDDD0004
A[1]	0xDDDD0008
A[2]	0xDDDD000C
...	...
A[7]	0xDDDD0020
A[8]	0xDDDD0024
A[9]	0xDDDD0028

Pointer pada array sesungguhnya juga diperintahkan pointer agar menunjuk ke alamat yang telah dialokasikan oleh larik tersebut. Pada contoh diatas, bila didefinisikan suatu pointer.

```
int *P:
```

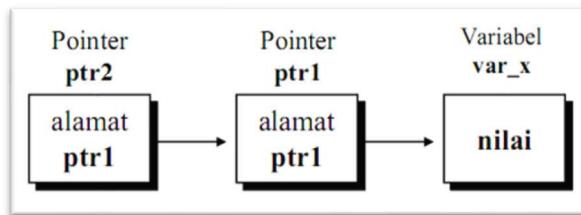
kemudian kita tunjuk ke alamat larik A

```
P =  
&A[0];
```

maka alamat P akan menunjuk ke alamat 0xDDDD0004.

5. Pointer menunjuk Pointer (Pointer to Pointer)

Suatu pointer bisa menunjuk ke pointer lain. Gambar berikut memberikan contoh mengenai pointer to pointer



- a. Untuk membentuk rantai pointer seperti gambar di atas, pendeklarasian yang diperlukan :

```
int var_x (variabel bertipe int)
```

```
int *ptr1; (variabel pointer yang menunjuk ke data bertipe int)
```

```
int **ptr2;(variabel pointer yang menunjuk ke pointer int)
```

- b. Agar ptr1 menunjuk ke variabel var_x, perintah yang diperlukan berupa:

```
ptr1 = &var_x;
```

- c. Sedangkan agar ptr2 menunjuk ke ptr1, instruksi yang diperlukan adalah

```
ptr2 = &ptr1;
```

LATIHAN

1. Struct – 1

Koreksi beberapa kesalahan dari listing program berikut sehingga program dapat dijalankan.

```
/* Nama File : struct1
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/
```

```

#include <stdio.h>
struct mahasiswa
{
    char nim [25];
    char nama [25];
    int usia;
};

struct mataKuliah
{
    char namamk [25];
    int semester;
    int sks;
};

void main ()
{
    struct mahasiswa mhs1 = {"1700001", "Ari Andana", 18};
    struct mataKuliah mk1 = {"Struktur Data", 2, 2};

    //tampilkan data mahasiswa
    printf ("NIM\t\t: %s\n", mhs1.nim);
    printf ("Nama Mahasiswa\t: %s\n", mhs1.nama_mhs);
    printf ("Usia Mahasiswa\t: %s\n\n", mhs.usia);

    //tampilkan data mata kuliah
    printf ("Mata Kuliah\t: %s\n", mk1.nama_mk);
    printf ("Semester\t: %d\n", mk1.semester);
    printf ("SKS\t\t: %d\n", mk1.ipk);
}

```

2. Struct – 2

Koreksi beberapa kesalahan dari listing program berikut sehingga program dapat dijalankan.

```

/* Nama File  : struct2
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

#include <stdio.h>

struct human {
    char nama [25];
    int usia;
};

```

```
struct pegawai{  
    char nip[7];  
    char jabatan[20];  
    struct human manusia;  
};  
  
int main (void)  
{  
    struct pegawai seseorang = {"010110", "Programmer",  
    {"Abdul Kadir", 22}};  
  
    printf ("Pegawai dengan NIP: %d  
    \nJabatan: %s  
    \nBernama: %s (%d tahun)\n",  
    seseorang.nip, seseorang.jabatan,  
    seseorang.manusia.nama,  
    seseorang.manusia.umur);  
  
    return 0;  
}
```

3. Struct – 3

Koreksi beberapa kesalahan dari listing program berikut sehingga program dapat dijalankan.

```
/* Nama File : struct3  
Pembuat : tuliskan nama dan NIM anda  
Tgl pembuatan : tuliskan tanggal hari ini*/  
  
#include <stdio.h>  
#include <string.h>  
  
typedef struct {  
    char tipe[5];  
    int masaUji;  
}machine;  
  
typedef struct {  
    char model[15];  
    char warna[10];  
    char bahanBakar[15];  
    machine mesin;  
}moto;  
  
void tampilInfo(moto);
```

```

void gantiBahanBakar(moto *);

int main(void)
{
    moto mx = {"Jupiter MX", "Merah", "Pertamax", {"DOHC",
5}};
    tampilInfo(mx);
    gantiBahanBakar(&mx);
    tampilInfo(mx);
    return 0;
}

void tampilInfo(moto items) {
    puts("Moto Info");
    puts("=====");
    printf("Model : %s (%s)\n"
           "Bahan Bakar : %s\n"
           "Tipe Mesin : %s\n"
           "Masa Uji Mesin : %d (tahun)\n",
           items.model, items.warna, items.bahanBakar,
           items.mesin.tipe, items.mesin.masaUji);
    puts("=====");
}

void gantiBahanBakar(moto *items) {
    printf("Bahan bakar motor saat ini : %s\n",
           items->bahanBakar);
    printf("Ketikkan bahan bakar yang ingin digunakan : ");
    fgets((*items).bahanBakar, 15, stdin);
    items->bahanBakar[strlen(items->bahanBakar) - 1] =
    '\0';
}

```

4. Pointer (Reference Operator)

```

/* Nama File : pointer1
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini */

```

```
#include <stdio.h>

main()
{
    int y, x = 87;           /* x & y bertipe int */
    int *px;
    /* var pointer yang menunjuk ke data yang bertipe int */

    px = &x;   /* px diisi dengan alamat dari variabel x */
    y = *px;  /* y diisi dengan nilai yg ditunjuk oleh px */

    printf("Alamat x      = %p\n", &x);
    printf("Isi px        = %p\n", px);
    printf("Isi x         = %d\n", x);
    printf("Nilai yang ditunjuk oleh px = %d\n", *px);
    printf("Nilai y         = %d\n", y);
}
```

5. Pointer (String menggunakan pointer)

```
/* Nama File : pointer2
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : Menukar isi 2 string dengan
                fasilitas pointer */

#include <stdio.h>
#include <string.h>

char *namal = "TEKNIK";
char *nama2 = "INFORMATIKA";

main ()
{
    char *namax;

    puts("OLD: ");
    printf("Nama 1 : %s\n", namal);
    //namal: pointer yang menunjuk ke string TEKNIK
    printf("Nama 2 : %s\n", nama2);
    //nama2: pointer yang menunjuk ke string INFORMATIKA

    namax = namal;
    namal = nama2;
    nama2 = namax;

    puts ("NOW: ");
    printf("Nama 1 : %s\n", namal);
```

```

        printf("Nama 2 : %s\n", nama2);
    }
}

```

6. Pointer to Pointer

```

/* Nama File  : pointer3
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini
Deskripsi     : contoh program untuk pointer menunjuk
pointer */

```

```

#include <stdio.h>

main()
{
    int var_x = 273;
    int *ptr1;
    int **ptr2;

    ptr1 = &var_x;
    ptr2 = &ptr1;

    printf("Nilai var_x = %d\n", *ptr1);
    printf("Nilai var_x = %d\n", **ptr2);
}

```

TUGAS

1. Buatlah struct untuk data lagu yang berisi tentang judul lagu, penyanyi, tahun produksi, nomor track dan kode album.
 2. Ketentuan : program ini akan memiliki dua buah struct, yaitu struct lagu dan struct kodeRBT. Jumlah data yang diinputkan dinamis (maks. 20 lagu)
- Perbaiki beberapa kesalahan pada program:

```

#include <stdio.h>

main()
{
    int *ptr;
    int k;
    k=7;
    printf("Isi variabel k\t: %d",K);
    printf("\nAlamat variabel k\t: %d",&k);
    printf("\nAlamat variabel *ptr\t: %d",&ptr);
    printf("\nIsi variabel *ptr\t: %d",*ptr);
    ptr=&k;
}

```

```
    printf("\n\nAlamat variabel *ptr\t: %f",&ptr);
    printf("\nIsi variabel *ptr\t: %f",ptr);
    printf("\nIsi dari alamat %d\t: %f",ptr,*ptr);
    printf("\n");
}
```

3. Buat program untuk menampilkan sebaris string seperti contoh berikut:
“Pendidikan Teknik Informatika”

Menggunakan variabel pointer (Pointer to String)

DAFTAR PUSTAKA

Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.

Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*. Bandung : Unikom.

Liem, Ingriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.

Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.

Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com