16TIN2074 – Struktur Data dan Algoritma Praktek

Case Study 14 Program (Alokasi Dinamis)

Dikerjakan oleh:

Muhammad Azhar Alauddin – 201524013

1AD4 Jurusan Teknik Komputer dan Informatika

Tugas ini dikumpulkan untuk memenuhi sebagian persyaratan kelulusan mata kuliah Struktur Data dan Algoritma Praktek

Program Studi D4 Teknik Informatika

Jurusan Teknik Komputer dan Informatika

Politeknik Negeri Bandung

2020/2021

Program PTR1.CPP

1. Implementasi Program

```c
/*---------------------------
File Program : PTR1.cpp
Contoh pemakaian pointer
Tujuan : Mengetahui panjang dan posisi variabel di memory
---------------------------*/

#include <stdio.h>

int main()
{
  int x, y; //deklarasi variable bertipe integer
  int *px;  //deklarasi variable bertipe pointer
  x = 87;  //pengisian variabel bertipe integer
  px = &x;  //pengisian variabel bertipe alamat
  y = *px;  //pengisian variabel bertipe integer dari alamat yang ditunjuk pointer
  printf("Alamat x = %p\n", &x); //menampilkan alamat x
  printf("Isi px = %d\n", x);  //menampilkan variabel x yang ditunjuk pointer px
  printf("Nilai yang ditunjuk oleh px = %d\n", *px); // menampilkan value yang ada
 dalam suatu alamat
  printf("Nilai y = %d\n", y); //menampilkan variable y

  return 0;
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Alamat x = 000000000061FE10
Isi px = 87
Nilai yang ditunjuk oleh px = 87
Nilai y = 87
```

- Tambahkan pernyataan untuk menampilkan Alamat varabel y, eksekusi, dan perhatikan hasilnya

```
Alamat x = 000000000061FE14
Isi px = 87
Nilai yang ditunjuk oleh px = 87
Nilai y = 87
Alamat y 000000000061FE10
```

- Bila pernyataan px = &x; diganti dengan y = *px; ,apa yang terjadi?
  Program akan error, karena tidak ada value berupa alamat pada px

- **Tuliskan hasil pemahaman Anda :**
  Setiap variable memiliki alamat yang kemudian bisa diakses dan didapatkan dengan operator ampersand(&). Dan value dari alamat yang ditunjuk bisa diakses dengan menggunakan operator(*)

Program PTR2.CPP

1. Implementasi Program

```c
/*---------------------------
File Program : PTR2.cpp
Contoh pemakaian pointer yang salah
Tujuan : Mengetahui panjang dan posisi variabel di memory
---------------------------*/

#include <stdio.h>

int main()
{
  float *pu;               //deklarasi variable pointer bertipe float
  float nu;                //deklarasi variable yang bertipe float
  int u = 1234;            //pengisian variable bertipe integer
  pu = (float *)&u;        //pengisian alamat dari variable u bertipe float
  nu = *pu;                //pengisian value nu dengan pointer pu
  printf("u = %d\n", u);   //menampilkan variable u bertipe integer
  printf("nu = %f\n", nu); //menampilkan variable nu bertipe float

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
u = 1234
nu = 0.000000
```

- **Tuliskan hasil pemahaman Anda :**
  Jika ingin melakukan pengisian(assignment), maka tipe dari variable pointer dan variable yang terkait harus sama. Semisal dalam contoh di atas, maka untuk menyamakan tipe, harus menggunakan (float *)

Program PTR3.CPP

1. Implementasi Program

```c
/*---------------------------
File Program : PTR3.cpp
Contoh pengubahan isi suatu variabel melalui pointer
Tujuan : Menggunakan type data pointer dan manipulasinya
----------------------------*/

#include <stdio.h>

int main()
{
  float d, *pd; //deklarasi variable bertipe float dan variable pointer bertipe float
  d = 54.5; //pengisian variable d
  printf("Isi d smula = %g\n", d); //menampilkan value variable d sebelum dilakukan operasi
  pd = &d; //pengisian alamat variable pada variable pointer
  *pd = *pd + 10; //operasi pertambahan value yang ditunjuk pointer
  printf("Isi d kini = %g\n", d); //menampilkan value dari variable d setelah di lakukan operasi
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Isi d semula = 54.5
Isi d kini = 64.5
```

- **Tuliskan hasil pemahaman Anda :**
  Pointer *pd menunjuk value yang ada pada variable d dan diisi dengan alamat d. Ini mengakibatkan jika pd dikenakan operasi, maka variable d pun akan berubah juga. Sehingga ketika variable d ditampilkan, maka akan sama dengan value yang ada pada variable pd

Program PTR4.CPP

1. Implementasi Program

```
/*---------------------------
File Program : PTR4.cpp
Contoh operasi pemakaian pointer
Tujuan : Melakukan operasi pada nilai yang ditunjuk pada pointer
----------------------------*/

#include <stdio.h>

int main()
{
  int z, s, *pz, *ps; //deklarasi variable bertipe integer dan variable pointer
bertipe integer
  z = 20; //pengisian variable z bertipe integer
  s = 30; //pengisian variable s bertipe integer
  pz = &z; //pengisian alamat pada variable pointer pz yang bertipe integer
  ps = &s; //pengisian alamat pada variable pointer ps yang bertipe integer
  *pz = *pz + *ps; //operasi pertambahan value yang ditunjuk pointer pz oleh val
ue yang ditunjuk pointer pz dan ps
  printf("z = %d, s = %d\n", z, s); //menampilkan value dari variable z dan s

  return 0;
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
z = 50, s = 30
```

- Modifikasilah pernyataan di atas, agar nilai Z (20) tidak ter-replace dengan nilai hasil operasi pertambahan. Pengubahan dilakukan sedemikian rupa sehingga terdapat nilai Z, S dan hasil pertambahannya.
- **Tuliskan hasil pemahaman Anda :**
  Pointer *pz menunjuk value yang ada pada variable z dan diisi dengan alamat z. Ini mengakibatkan jika pz dikenakan operasi, maka variable z pun akan berubah juga. Solusi dari permasalahan ini adalah dengan membuat variable baru, seperti contoh disini dengan menggunakan variable result.

Program PTR5.CPP

1. Implementasi Program

```
/*---------------------------
File Program : PTR5.cpp
Contoh pointer ke type dasar, mendeklarasikan & alokasi variabel dinamik
Tujuan : Memahami operasi pada pointer
---------------------------*/

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

int main()
{
  // Kamus Data (deklarasi, inisialisasi nilai variabel)
  int i = 5, j; // Pendeklarasian dan Pengisian nilai integer
  char c = 'X';  // Pengisian nilai character
  int *Ptri = (int *)malloc(4);  // Pengalokasian memori untuk pointer Ptri
  int *Ptrj = (int *)malloc(sizeof(int));  // Pengalokasian memori untuk pointer Ptr
j
  float *fx = (float *)malloc(sizeof(float));  // Pengalokasian memori untuk pointer
 fx
  int A[5]; // Pendeklarasian Array A dengan jumlah indeks sebesar 5
  float f = 7.23;  // Pendeklarasian dan Pengisian nilai float

  // Program
  *Ptri = 8;  // Pengisian nilai integer ke Pointer Ptri
  *Ptrj = 0;  // Pengisian nilai integer ke Pointer Ptrj
  *fx = 3;  // Pengisian nilai integer ke Pointer fx
  printf("Alamat i = %x\n", &i);  // Menampilkan alamat i
  printf("Nilai i = %d\n", i);  // Menampilkan nilai i
  printf("Ukuran int = %d byte\n\n", sizeof(int));  // Menampilkan ukuran memori dar
i integer

  printf("Alamat j = %x\n", &j); // Menampilkan alamat j
  printf("Nilai j = %d\n", j);  // Menampilkan nilai j

  printf("Alamat c = %x\n", &c); // Menampilkan alamat c
  printf("Nilai c = %c\n", c);  // Menampilkan nilai c
  printf("Ukuran char = %d byte\n\n", sizeof(char)); // Menampilkan ukuran memori da
ri char

  printf("Alamat Ptri = %x\n", Ptri); // Menampilkan alamat pointer Ptri
  printf("Isi var Ptri = %x\n", Ptri); // Menampilkan isi dari Ptri
```

```c
  printf("Nilai yang ditunjuk oleh Ptri = %d\n", *Ptri); // Menampilkan nilai yang d
itunjuk oleh pointer Ptri
  printf("Ukuran pointer int = %d byte\n\n", sizeof(int *)); // Menampilkan ukuran m
emori dari pointer integer

  printf("Alamat Ptrj = %x\n", &Ptrj); // Menampilkan alamat pointer Ptrj
  printf("Isi var Ptrj = %x\n", Ptrj); // Menampilkan isi dari Ptrj
  printf("Nilai yang ditunjuk oleh Ptrj = %d\n", *Ptrj); // Menampilkan nilai yang d
itunjuk oleh pointer Ptrj

  Ptrj = Ptri; // Mengisi nilai Ptrj dengan Ptri
  printf("Isi var Ptrj sekarang = %x\n", Ptrj); // Menampilkan isi dari Ptrj
  printf("Nilai yang ditunjuk oleh Ptrj sekarang = %d\n", *Ptrj); // Menampilkan nil
ai yang ditunjuk oleh pointer Ptrj setelah perubahan

  printf("Alamat A = %x\n", &A); // Menampilkan alamat A
  printf("Isi var A = %x\n", A[0]); // Menampilkan isi dari A
  printf("Ukuran array A = %d byte\n\n", sizeof(A)); // Menampilkan ukuran memori da
ri A

  printf("Alamat f = %x\n", &f); // Menampilkan alamat f
  printf("Nilai f = %f\n", f); // Menampilkan nilai dari f
  printf("Ukuran float = %d byte\n\n", sizeof(float)); // Menampilkan ukuran memori
dari float

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Alamat i = 61fe0c
Nilai i = 5
Ukuran int = 4 byte

Alamat j = 61fe08
Nilai j = 51
Alamat c = 61fe07
Nilai c = X
Ukuran char = 1 byte

Alamat Ptri = 6e1420
Isi var Ptri = 6e1420
Nilai yang ditunjuk oleh Ptri = 8
Ukuran pointer int = 8 byte
```

```
Alamat Ptrj = 61fdf8
Isi var Ptrj = 6e1440
Nilai yang ditunjuk oleh Ptrj = 0
Isi var Ptrj sekarang = 6e1420
Nilai yang ditunjuk oleh Ptrj sekarang = 8
Alamat A = 61fde0
Isi var A = 0
Ukuran array A = 20 byte

Alamat f = 61fddc
Nilai f = 7.230000
Ukuran float = 4 byte
```

- **Tuliskan hasil pemahaman Anda :**
  Isi dari variabel pointer satu dapat dipindahtangankan ke variabel pointer lain. Dan ketika sudah dipindahtangankan, maka nilai yang ditunjuk oleh kedua variabel pointer tersebut adalah sama. Dalam hal ini, sama seperti variabel Ptrj dan juga Ptri. Selain itu, setiap variable pointer yang belum menunjuk variabel apapun harus dialokasikan terlebih dahulu di memori. Tambahan, tipe data memiliki ukuran memorinya masing-masing. Float(4 byte), int(4 byte), Pointer Int(8 byte), char(1 byte), dan Array tergantung dengan isi dari Array-nya itu sendiri seperti dalam contoh Array berisi 5 indeks integer, maka ukuran memorinya adalah sebesar 4 x 5 byte = 20 byte

Program PTR6.CPP

1. Implementasi Program

```
/*----------------------------
File Program : PTR6.cpp
Contoh pointer menunjuk ke array
Tujuan : Memahami operasi pointer yang menunjuk kepada array
----------------------------*/

#include <stdio.h>

int main()
{
  // Kamus Data(Deklarasi, inisialisasi nilai variabel)
  static int tgl_lahir[] = {18, 6, 1989};
  int *ptgl; // Pendeklarasian pointer ptgl

  ptgl = tgl_lahir; // Pengisian ptgl dengan alamat dari Array tgl_lahir
  printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl); // Menampilkan nilai ya
ng ditunjuk oleh ptgl
  printf("Nilai dari tgl_lahir[0] = %d\n", tgl_lahir[0]); // Menampilkan nilai a
rray tgl _lahir dengan indeks ke-0

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Nilai yang ditunjuk oleh ptgl = 18
Nilai dari tgl_lahir[0] = 18
```

- Menampilkan seluruh elemen pada Array tgl_lahir

```
ptgl = tgl_lahir;
printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
ptgl += 1;
printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
ptgl += 1;
printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
```

```
Nilai yang ditunjuk oleh ptgl = 18
Nilai yang ditunjuk oleh ptgl = 6
Nilai yang ditunjuk oleh ptgl = 1989
```

- **Tuliskan hasil pemahaman Anda :**

  Untuk mengambil alamat dari suatu Array, maka kita hanya perlu menuliskan nama dari variable Array tersebut(tidak perlu menggunakan Ampersand(&)). Dan untuk menampilkan seluruh elemen yang ada pada Array, maka kita hanya perlu mengubah alamat dari pointer/variable yang ditunjuk pointer tersebut dengan menambah 1 (+4 byte) untuk berpindah ke indeks selanjutnya.

Program PTR7.CPP

1. Implementasi Program

```
/*---------------------------
File Program : PTR7.cpp
Contoh pointer dan string bagian ke-1
Tujuan : Memahami operasi pointer yang menunjuk ke data string
----------------------------*/

#include <stdio.h>

int main()
{
  // Kamus Data(Deklarasi, inisialisasi nilai variabel)
  char *pkota = "BANDUNG"; // Pendeklarasian dan Pengisian suatu variabel dengan
 tipe character

  puts(pkota); // Menampilkan variable dengan menggunakan fungsi puts()

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
PTR7.cpp:12:17: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]
   char *pkota = "BANDUNG";
                 ^~~~~~~~~
BANDUNG
```

- Perbedaan char *pkota=”BANDUNG” dengan char kota[]=”BANDUNG”
  Dengan mendeklarasikan char *pkota = “BANDUNG”, maka yang dideklarasikan adalah berupa **Character**, sedangkan jika mendeklarasikan dengan char kota[] = “BANDUNG” maka yang dideklarasikan adalah berupa **Array of Character** atau **String**
- **Tuliskan hasil pemahaman Anda :**
  Dengan fungsi puts(), kita dapat menampilkan nilai yang ditunjuk oleh pointer tanpa harus menggunakan tanda asterisk(*). Dan untuk mendeklarasikan Array of Character, sebaiknya menggunkan tanda kurung siku [].

Program PTR8a.CPP

1. Implementasi Program

```
/*---------------------------
File Program : PTR8a.cpp
Contoh Pointer dan String bagian ke-2
Tujuan : Memahami operasi menukarkan isi dua buah string tanpa pemakaian pointer
----------------------------*/

#include <stdio.h>
#include <string.h>
#define PANJANG 20 // Men-define variable PANJANG dengan 20

int main()
{
  // Kamus Data(Deklarasi, inisialisasi nilai variabel)
  char nama1[PANJANG] = "DEWI SARTIKA"; // Pendeklarasian dan Pengisian variabel berti
pe Array Of Character
  char nama2[PANJANG] = "SULTAN HASANUDDIN"; // Pendeklarasian dan Pengisian variabel
bertipe Array Of Character
  char namaX[PANJANG]; // Pendeklarasian variabel bertipe Array Of Character

  puts("Nama semula : "); // Menampilkan suatu format/teks dengan fungsi puts
  printf("Nama 1 --> %s\n", nama1); // Menampilkan variabel bertipe Array of Character
  printf("Nama 2 --> %s\n", nama2); // Menampilkan variabel bertipe Array of Character

  strcpy(namaX, nama1); // Menyalin isi dari String nama1 ke namaX
  strcpy(nama1, nama2); // Menyalin isi dari String nama2 ke nama1
  strcpy(nama2, namaX); // Menyalin isi dari String namaX ke nama2

  puts("Nama sekarang : "); // Menampilkan suatu format/teks dengan fungsi puts
  printf("Nama 1 --> %s\n", nama1); // Menampilkan variabel bertipe Array of Character
  printf("Nama 2 --> %s\n", nama2); // Menampilkan variabel bertipe Array of Character

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Nama semula :
Nama 1 --> DEWI SARTIKA
Nama 2 --> SULTAN HASANUDDIN
Nama sekarang :
Nama 1 --> SULTAN HASANUDDIN
Nama 2 -- > DEWI SARTIKA
```

- Tuliskan hasil pemahaman Anda :
  Penyalinan dari satu String ke String lain, kita bisa menggunakan fungsi strcpy dengan menambahkan header <string.h>. Selain menggunakan fungsi printf() untuk menampilkan format ataupun variabel, kita juga bisa menggunakan fungsi puts(). Untuk pendeklarasian dan pengisian nilai dari suatu variabel, kita bisa menggunakan #define_(variabel)_(nilai).

Program PTR8b.CPP

1. Implementasi Program

```c
/*---------------------------
File Program : PTR8b.cpp
Contoh Pointer dan String bagian ke-3
Tujuan : Memahami operasi menukarkan isi dua buah string yang ditunjuk oleh pointer
---------------------------*/

#include <stdio.h>

int main()
{
  // Kamus Data(Deklarasi, inisialisasi nilai variabel)
  char *nama1 = "DEWI SARTIKA"; // Pendeklarasian dan Pengisian variabel bertipe Array Of Character
  char *nama2 = "SULTAN HASANUDDIN"; // Pendeklarasian dan Pengisian variabel bertipe Array Of Character
  char *namaX; // Pendeklarasian variabel bertipe Array Of Character
  puts("Nama semula : "); // Menampilkan suatu format/teks dengan fungsi puts
```

```
  printf("Nama 1 --> %s\n", nama1); // Menampilkan variabel bertipe Array of Character
  printf("Nama 2 --> %s\n", nama2); // Menampilkan variabel bertipe Array of Character

  // Pertukaran string
  namaX = nama1; // Pengisian namaX dengan address nama1
  nama1 = nama2; // Pengisian nama1 dengan address nama2
  nama2 = namaX; // Pengisian nama2 dengan address namaX

  puts("Nama sekarang : "); // Menampilkan suatu format/teks dengan fungsi puts
  printf("Nama 1 --> %s\n", nama1); // Menampilkan variabel bertipe Array of Character
  printf("Nama 2 --> %s\n", nama2); // Menampilkan variabel bertipe Array of Character

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Nama semula :
Nama 1 --> DEWI SARTIKA
Nama 2 --> SULTAN HASANUDDIN
Nama sekarang :
Nama 1 --> SULTAN HASANUDDIN
Nama 2 -- > DEWI SARTIKA
```

- **Tuliskan hasil pemahaman Anda :**
  Untuk melakukan pertukaran isi nilai dari suatu variabel bertipe Array, bisa dengan saling meng-*assign* alamat dari masing-masing Array-nya.

Program PTR9.CPP

1. Implementasi Program

```
/*----------------------------
File Program : PTR9.cpp
Contoh Pointer yang menunjuk ke pointer
Tujuan : Memahami operasi pointer yang menunjuk ke pointer
----------------------------*/
```

```c
#include <stdio.h>

int main()
{
  // Kamus Data(Deklarasi, inisialisasi nilai variabel
  int var_x = 273;
  int *ptr1;
  //ptr1 adl pointer yang menunjuk ke objek bertipe integer
  // Boleh disebut pointer integer saja

  int **ptr2;
  //ptr2 adl pointer yang menunjuk ke pointer yang menunjuk ke objek
  // Bertipe integer(Boleh disebut : Pointer menunjuk ke pointer integer saja)

  ptr1 = &var_x;
  ptr2 = &ptr1;

  // Mengakses nilai var_x melalui ptr1 dan ptr2
  printf("Nilai var_x = %d\n", *ptr1);
  printf("Nilai var_x = %d\n", **ptr2);

  return 0;
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Nilai var_x = 273
Nilai var_x = 273
```

- Tuliskan hasil pemahaman Anda :
  Simbol Asterisk(*) menunjukkan jumlah maksimal pointer dalam menunjuk suatu variabel ataupun pointer lainnya lebih dari satu kali. Seperti pada contoh, **ptr2. Artinya pointer ptr2 dapat menunjuk ke variabel ataupun pointer lainnya lebih dari satu kali.

Program PTR10.CPP

1. Implementasi Program

```
/*---------------------------
File Program : PTR10.cpp
Contoh Pointer yang menunjuk ke prosedur
Tujuan : Memahami operasi pointer yang menunjuk ke prosedur
---------------------------*/

#include <stdio.h>

// Prototype
void f1(void);
void f2(void);
void f3(void);
void f4(void);

// Kamus Global
#define true 1
#define false 0
int quit = false;

int main()
{
  // Kamus Lokal
  // Definisi tabel yang elemennya adalah pointer ke fungsi
  // Elemen tabel yang ke - i akan mengakses fungsi ke - i
  // Pilihan menjadi indeks tabel, dan dipakai untuk mengaktifkan fungsi yang sesuai

  void (*tab[4])() = {f1, f2, f3, f4}; // Pointer ke Procedure

  // Program
  printf("Pointer to Function : \n");

  // Menu
  do
  {
    printf("Pilih salah satu : \n");
    printf("1. Buka file hanya untuk baca \n");
    printf("2. Tutup file \n");
    printf("3. Edit file \n");
    printf("4. Quit \n");

    tab[getchar() - '1']();
```

```
    getchar(); // untuk membuang return karakter
  } while (!quit);

  return 0;
}

// Body Function
void f1(void)
{
  printf("Ini prosedur f1 \n");
}

void f2(void)
{
  printf("Ini prosedur f2 \n");
}

void f3(void)
{
  printf("Ini prosedur f3 \n");
}

void f4(void)
{
  printf("Ini prosedur f4 \n");
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Pointer to Function :
Pilih salah satu :
1. Buka file hanya untuk baca
2. Tutup file
3. Edit file
4. Quit
1
Ini prosedur f1
Pilih salah satu :
1. Buka file hanya untuk baca
2. Tutup file
3. Edit file
4. Quit
2
```

```
Ini prosedur f2
Pilih salah satu :
1. Buka file hanya untuk baca
2. Tutup file
3. Edit file
4. Quit
```

- Tuliskan hasil pemahaman Anda :
  Pointer dapat berisi alamat dari suatu *procedure* atau *function*. Tipe data pointer juga harus sama dengan tipe return dari *procedure* atau *function* yang akan dirujuk

Program PTR11.CPP

1. Implementasi Program

```
/*----------------------------
File Program : PTR11.cpp
Contoh Pointer yang menunjuk ke fungsi yang memiliki parameter
Tujuan : Memahami operasi pointer yang menunjuk ke fungsi. Melakukan
offset terhadap tabel tergantung fungsi f
----------------------------*/

#include <stdio.h>

// Prototype
// Prototype, fungsi yang diberikan sebagai parameter aktual

int succ(int i); // succ -> singkatan dari suksessor
int pred(int i); // pred -> singkatan dari predesessor

// Prosedur dengan parameter sebuah fungsi
void geser(int *TT, int (*f)(int));

// Kamus Global
int N; // Ukuran Efektif

int main()
{
  // Kamus Lokal
  int MyTab[10];
  int i;

  // Program
  N = 10;
  for (i = 0; i < N; i++)
  {
    MyTab[i] = i;
```

```c
  }

  printf("\nIsi tabel dalam main sebelum pemanggilan : \n");

  for (i = 0; i < N; i++)
  {
    printf(" %d ", MyTab[i]);
  }

  printf("\n");

  // Pakai geser dengan parameter succ
  printf("\nGeser dengan succ \n");
  geser(MyTab, succ);
  printf(" dalam main \n");

  for (i = 0; i < N; i++)
  {
    printf(" %d ", MyTab[i]);
  }
  printf("\n");

  // Pakai geser dengan parameter pred
  printf("\nGeser dengan pred \n");
  geser(MyTab, pred);
  printf(" dalam main setelah aplikasi pred \n");
  for (i = 0; i < N; i++)
  {
    printf(" %d ", MyTab[i]);
  }
  printf("\n");
  return 0;
}

// Body Function
int succ(int i)
{
  return (i + 1);
}

int pred(int i)
{
  return (i - 1);
}
```

```c
void geser(int *TT, int (*f)(int))
{
  // Kamus Lokal
  int i;

  // Program
  printf(" dalam geser \n");
  for (i = 0; i < N; i++)
  {
    TT[i] = f(TT[i]);
    printf(" %d ", TT[i]);
  }
  printf("\n");
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Isi tabel dalam main sebelum pemanggilan :
 0  1  2  3  4  5  6  7  8  9

Geser dengan succ
 dalam geser
 1  2  3  4  5  6  7  8  9  10
 dalam main
 1  2  3  4  5  6  7  8  9  10

Geser dengan pred
 dalam geser
 0  1  2  3  4  5  6  7  8  9
 dalam main setelah aplikasi pred
 0  1  2  3  4  5  6  7  8  9
```

- Tuliskan hasil pemahaman Anda :
  Parameter harus dicantumkan pada saat pendeklarasian suatu pointer beserta tipe data nya. Contoh : void geser(int *TT, int (*f) (int)).

Program PTR12.CPP

1. Implementasi Program

```
/* ------------------------------------------------------- */
/* File Program : PTR12.CPP */
/* Contoh Pointer yang menunjuk ke Procedure dengan parameter input/output*/
/* Tujuan : memahami operasi pointer yang menunjuk ke prosedur yang memiliki
parameter. Melakukan offset terhadap tabel tergantung fungsi f */
/* ------------------------------------------------------- */

#include <stdio.h>

/* Prototype */
/* Prototype, fungsi yang diberikan sebagai parameter aktual */
void succ(int *i); /* suksessor, berupa procedure by ref */
void pred(int *i); /* predesessor */

/* Prosedur dengan parameter sebuah fungsi */
void geser(int *TT, void (*f)(int *));

/* kamus Global */
int N; /* ukuran efektif */
int main()
{
  /* kamus lokal */
  int MyTab[10];
  int i;
  /* program */
  N = 10;
  for (i = 0; i < N; i++)
    MyTab[i] = i;
  printf("Isi tabel dalam main sebelum pemanggilan : \n");
  for (i = 0; i < N; i++)
    printf(" %d ", MyTab[i]);
  printf("\n");

  /* Pakai geser dengan parameter succ */
  printf("Geser dengan succ \n");
  /* Memanggil modul geser dan menjadikan alamat array MyTab
 dan alamat dari modul int succ(int*) sebagai parameter. */
  geser(MyTab, succ);
  printf(" dalam main \n");
  for (i = 0; i < N; i++)
    printf(" %d ", MyTab[i]);
```

```
  printf("\n");

  /* Pakai geser dengan parameter pred */
  printf("Geser dengan pred \n");
  /* Memanggil modul geser dan menjadikan alamat array MyTab
 dan alamat dari modul int pred(int*) sebagai parameter. */
  geser(MyTab, pred);
  printf(" dalam main setelah aplikasi pred \n");
  for (i = 0; i < N; i++)
    printf(" %d ", MyTab[i]);
  printf("\n");

  return 0;
}
/*Body Function */
void succ(int *i)
{
  *i = *i + 1;
}
void pred(int *i)
{
  *i = *i - 1;
}
void geser(int *TT, void (*f)(int *))
{
  /* Kamus Lokal */
  int i;
  /*Program*/
  printf(" dalam geser \n");
  for (i = 0; i < N; i++)
  {
    /* Menjalankan fungsi f dengan passing alamat dari TT[i], fungsi
  f adalah parameter kedua dari modul geser. */
    f(&TT[i]);
    printf(" %d ", TT[i]);
  }
  printf("\n");
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ini

```
Isi tabel dalam main sebelum pemanggilan :
 0  1  2  3  4  5  6  7  8  9
Geser dengan succ
 dalam geser
 1  2  3  4  5  6  7  8  9  10
 dalam main
 1  2  3  4  5  6  7  8  9  10
Geser dengan pred
 dalam geser
 0  1  2  3  4  5  6  7  8  9
 dalam main setelah aplikasi pred
 0  1  2  3  4  5  6  7  8  9
```

- Tuliskan hasil pemahaman Anda :
  Pada variable pointer yang merujuk suatu modul(prosedur ataupun fungsi) dapat membawa parameter pointer juga dengan menambahkan asterisk(*) pada penulisan parameternya. Sebagai contoh int(*Modul)(int,char,char*)

Program PTR13.CPP

1. Implementasi Program

```c
/* ------------------------------------------------------------ */
/* File Program : PTR13.CPP */
/* Contoh Pointer yang menunjuk ke Procedure dengan parameter input/output*/
/* Tujuan : memahami operasi pointer yang menunjuk ke prosedur yang memiliki
parameter. Melakukan offset terhadap tabel tergantung fungsi f */
/* ------------------------------------------------------------ */

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

/* Prototype */
/* Prototype, fungsi yang diberikan sebagai parameter aktual */
void succI(void *i); /* suksessor, berupa procedure by ref */
void predI(void *i); /* predesessor */
void succC(void *c); /* suksessor, berupa procedure by ref */
void predC(void *c); /* predesessor */

/* Prosedur dengan parameter sebuah fungsi */
void geser(int *TT, void (*f)(void *));
/* print tabel yang belum ketahuan typenya */
```

```
void printtab(void *T, enum MyType Ty);

/* kamus Global */
int N; /* ukuran efektif */
enum MyType
{
  bulat,
  karakter
};
int main()
{
  void *MyTabInt;
  void *MyTabC;
  int i;
  /* program */
  N = 10;
  MyTabInt = (int *)malloc(N * sizeof(int));
  MyTabC = (char *)malloc(N * sizeof(char));
  /* assignment pointer bertipe data void* harus dilakukan
 casting agar compiler tahu berapa byte yang harus
 diubah. */
  *(int *)MyTabInt = 1;
  for (i = 0; i < N; i++)
  {
    *(int *)((int *)MyTabInt + i) = i;
    *(char *)((char *)MyTabC + i) = 'Z';
  }
  printf("Isi tabel dalam main sebelum pemanggilan : \n");
  printf(" Tabel integer \n");
  printtab((int *)MyTabInt, bulat);
  printf(" Tabel charakter \n");
  printtab((char *)MyTabC, karakter);
  printf("\n");
  /* Pakai geser dengan parameter succ */
  printf("Geser dengan succ \n");
  geser((int *)MyTabInt, succI);
  geser(reinterpret_cast<int *>(MyTabC), succC);
  printf(" dalam main \n");
  printf(" Tabel integer \n");
  printtab((int *)MyTabInt, bulat);
  printf(" Tabel charakter \n");
  printtab((char *)MyTabC, karakter);
  printf("\n");
  /* Pakai geser dengan parameter pred */
  printf("Geser dengan pred \n");
```

```c
  geser((int *)MyTabInt, predI);
  geser((int *)MyTabC, predC);
  printf(" dalam main \n");
  printf(" Tabel integer \n");
  printtab((int *)MyTabInt, bulat);
  printf(" Tabel charakter \n");
  printtab((char *)MyTabC, karakter);
  printf("\n");
  return 0;
}
/*Body Function */
void succI(void *i)
{
  *(int *)i = *(int *)i + 1;
}
void predI(void *i)
{
  *(int *)i = *(int *)i - 1;
}
void succC(void *c)
{
  *(char *)c = *(char *)c + 1;
}
void predC(void *c)
{
  *(char *)c = *(char *)c - 1;
}
void geser(int *TT, void (*f)(void *))
{
  /* Kamus Lokal */
  int i;
  /*Program*/
  printf(" dalam geser \n");
  for (i = 0; i < N; i++)
  {
    /* i bertambah sebanyak 4 byte karena TT bertipe data
 int*. Sehingga untuk penggeseran char* akan bergeser
 tiap 4 karakter (4 byte) */
    f(&TT[i]);
    printf(" %d ", TT[i]);
  }
  printf("\n");
}
void printtab(void *T, enum MyType Ty)
{
```

```
  int i;
  for (i = 0; i < N; i++)
  {
    switch (Ty)
    {
    case bulat:
      printf("%d ", (int *)*((int *)T + i));
      break;
    case karakter:
      printf("%c ", (char *)*((char *)T + i));
      break;
    }
  }
}
```

2. Case Study :

- Eksekusi file program tersebut dan pahami maksud program ininh

```
Isi tabel dalam main sebelum pemanggilan :
 Tabel integer
0 1 2 3 4 5 6 7 8 9  Tabel charakter
Z Z Z Z Z Z Z Z Z Z
Geser dengan succ
 dalam geser
 1  2  3  4  5  6  7  8  9  10
 dalam geser
 1515870811  1515870811  -1414833573  -1414812756  -17912916  -17891601  1  1  -
1076799210  402667305
 dalam main
 Tabel integer
1 2 3 4 5 6 7 8 9 10  Tabel charakter
[ Z Z Z [ Z Z Z [ Z
Geser dengan pred
 dalam geser
 0  1  2  3  4  5  6  7  8  9
 dalam geser
 1515870810  1515870810  -1414833574  -1414812757  -17912917  -17891602  0  0  -
1076799211  402667304
 dalam main
 Tabel integer
0 1 2 3 4 5 6 7 8 9  Tabel charakter
Z Z Z Z Z Z Z Z Z Z
```

- Tuliskan hasil pemahaman Anda :

Ukuran dari setiap satuan tipe data tentunya berbeda. Dan dari kasus PTR sebelumnya, bahwa pointer dapat bertipe data void(modul), maka disini agar ukuran dari pointer yang dideklarasikan bertipe void*, maka harus

ada casting terlebih dahulu agar jumlah byte yang berubah dapat ditentukan

LESSON LEARNED

Ada banyak sekali pelajaran yang saya bisa dapatkan, setelah mengerjakan tugas *Case Study* Pemahaman Alokasi Dinamis ini. Mungkin yang awalnya hanya tau dan mengerti sebatas teori mengenai pointer dan juga alokasi, tetapi dengan adanya tugas ini saya jadi mengetahui dan mengerti juga bagaimana secara praktisnya dalam implementasi algoritma. Berikut ringkasan dari beberapa hal yang saya baru pahami mengenai materi terkait :

- Pendeklarasian pointer yang menunjuk pada modul harus dideklarasikan beserta tipe data parameternya
- Pointer dapat berisi alamat suatu *procedure* atau *function*. Tipe data pointer harus sesuai dengan tipe return dari *procedure* atau *function*
- Array pada memori terletak saling bersebelahan, sehingga jika variable ptr merujuk ke array[0], maka ptr+1 akan merujuk ke array[1]
- Fungsi strcpy() dapat digunakan untuk menyalin isi dari variable bertipe string, tetapi tidak menyalin alamat rujukan dari suatu variabel string