Modul Praktikum

Alpro Lanjut

5

POINTER

DASAR TEORI

Pointer adalah sebuah variabel khusus dalam bahasa pemrograman C++ yang digunakan untuk menyimpan alamat memori dari suatu data atau variabel lain. Alamat memori ini merujuk ke lokasi di mana data tersebut disimpan di dalam memori komputer. Dengan menggunakan pointer, program dapat mengakses, memanipulasi, atau bahkan mengelola data secara langsung di tingkat memori, memberikan fleksibilitas dan efisiensi yang lebih besar dalam pengelolaan resources.

Pointer dideklarasikan dengan menambahkan tanda * (simbol asterisk) sebelum nama variabel, dan alamat memori suatu variabel dapat diperoleh menggunakan operator & (address-of). Untuk mengakses nilai yang disimpan di alamat memori yang ditunjuk oleh pointer, digunakan operator * (dereference).

Secara sederhana, pointer adalah alat yang memungkinkan programmer untuk bekerja langsung dengan memori, menjadikannya elemen penting dalam pemrograman tingkat rendah dan manajemen memori dinamis.

1. MEMORI & ALAMAT

Sebelum memahami tentang pointer, sebaiknya memahami dulu tentang **alamat dalam memori** komputer. Lokasi pada memori komputer itu memiliki alamat dan menyimpan sebuah nilai. Alamat atau address yang dimaksud bernilai numerik (umumnya dalam bentuk hexadecimal).

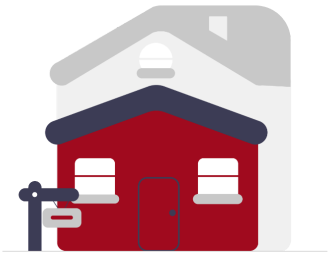
Setiap variabel yang dibuat pada program akan diberi lokasi di memori komputer. Nilai dari variabel sebenarnya disimpan pada alamat yang diberikan. Untuk mengetahui dimana data disimpan, pada C++ digunakan operator &.

Memori
Indonesia (Wilayah)

Address / Alamat
Rumah No 69, Jl. Wolvendom, Kecamatan Meikarta, Provinsi Kalimantan Tenggara

Variabel
rumahRidho

Value / Nilai
Ridho



2. PASS BY VALUE

Secara default di C++ semua variable itu **passing by value**. Bukan by reference. Artinya jika kita mengirim sebuah variabel ke dalam function, atau variable lain, sebenarnya yang dikirim adalah duplikasi value nya.

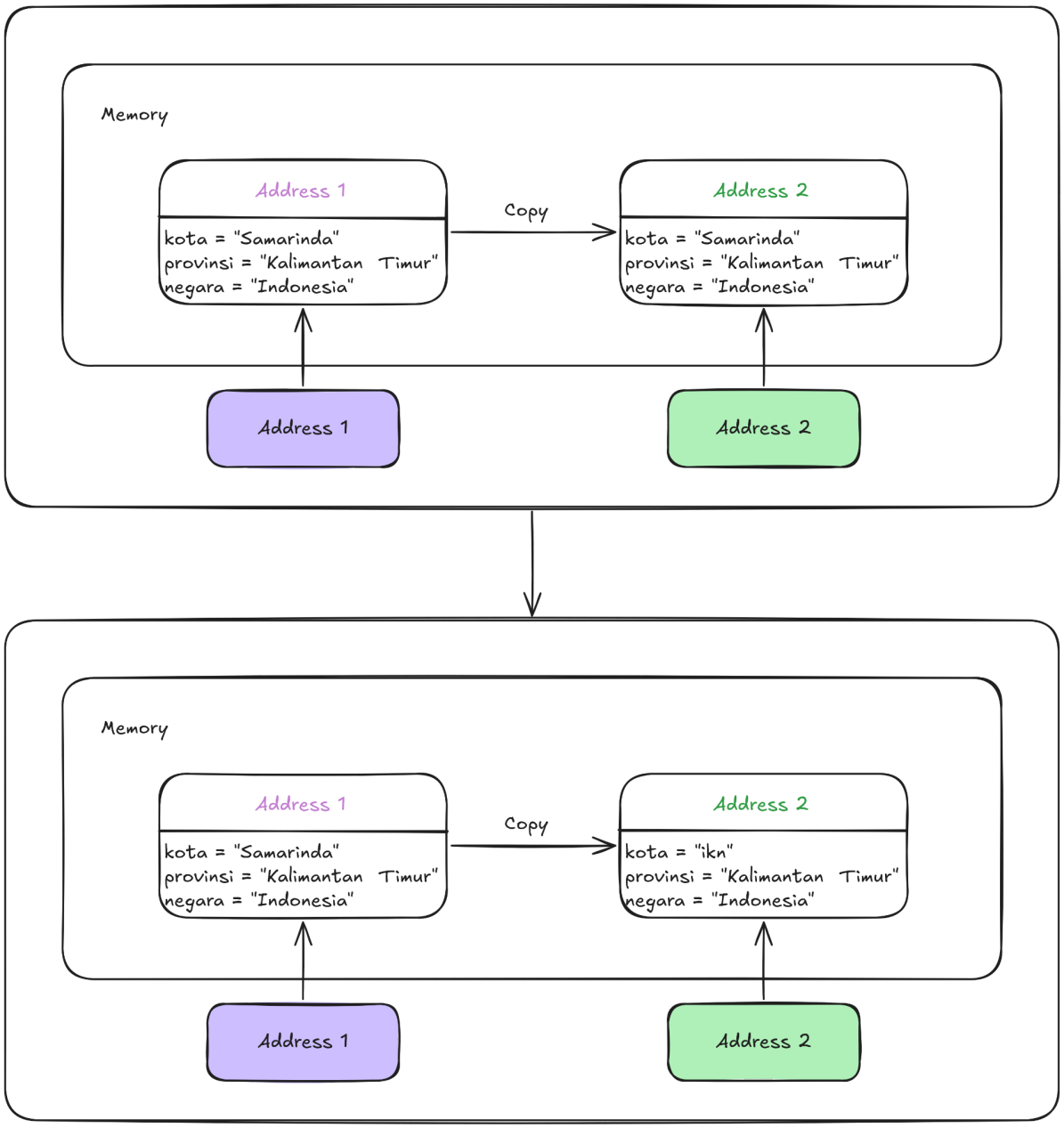
Contoh program pass by value:

```cpp
// passByValue.cpp
# include <iostream>
using namespace std;

struct Address{
    string kota;
    string provinsi;
    string negara;
};

int main(){
    // deklarasi variabel dengan struct
    Address address1, address2;
    // mengisi value address1
    address1.kota = "Samarinda";
    address1.provinsi = "Kalimantan Timur";
    address1.negara = "Indonesia";

    // mengcopy value address1 ke address2
    address2 = address1;

    // mengganti property kota dari samarinda ke ikn
    address2.kota = "ikn";

    // value address1
    cout<< address1.kota<<endl;
    // value address2
    cout<< address2.kota;
    return 0;
}
```

Output:

```
→ g++ passByValue.cpp ; ./a.exe
Samarinda
ikn
```

Visualisasi proses pengubahan alamat dengan contoh di atas



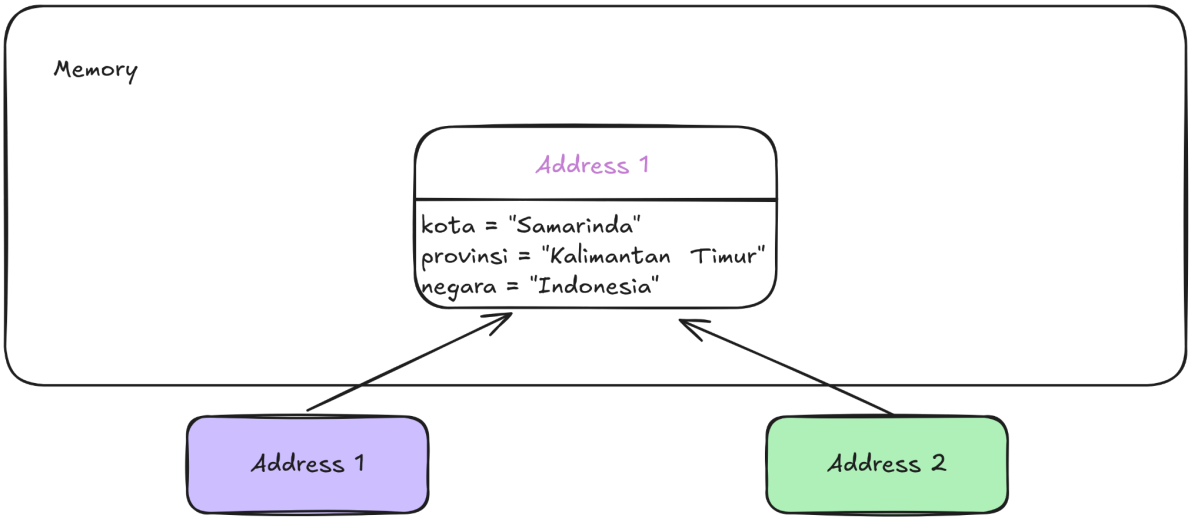
Terlihat bahwa address1 dan address2 menunjuk alamat yang berbeda.

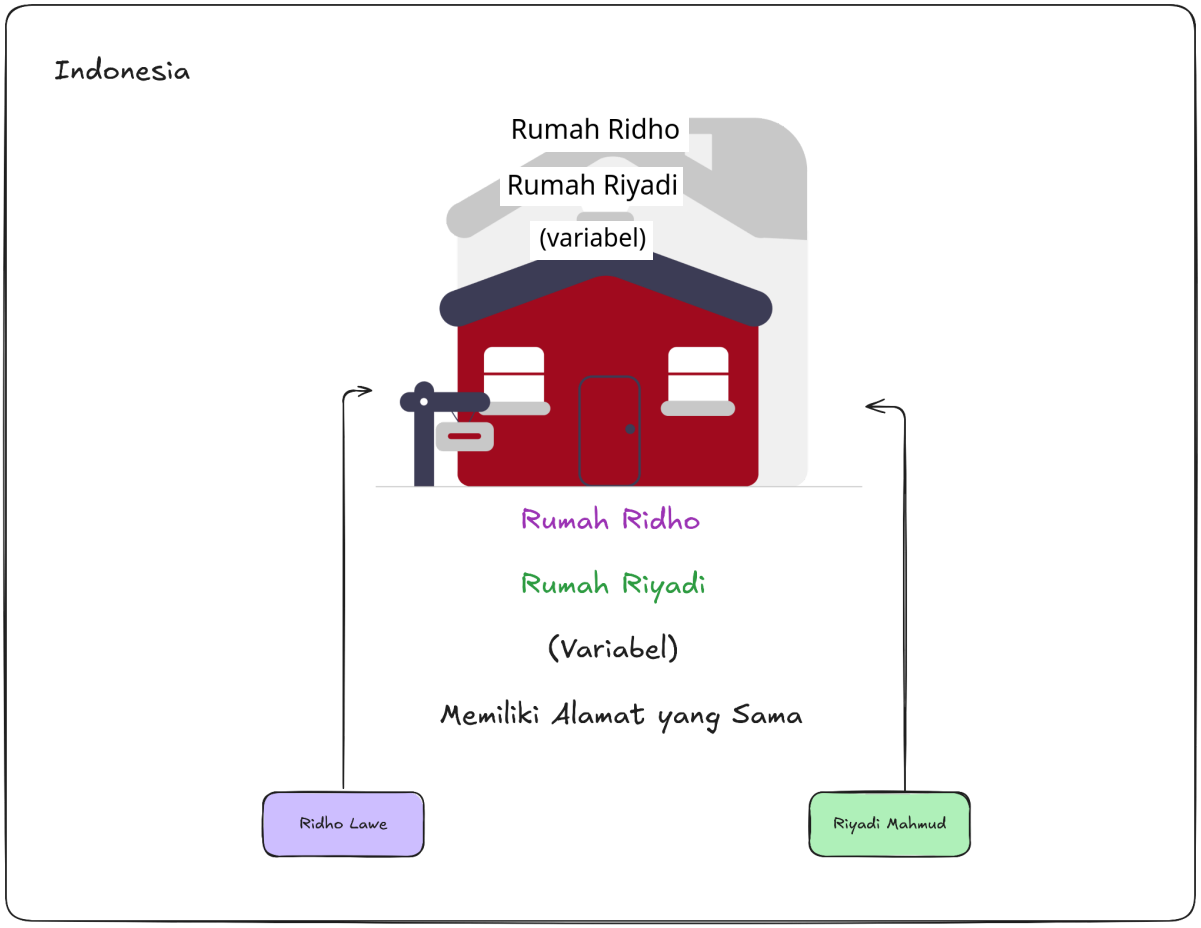
3. POINTER / MENGAKSES PASS BY REFERENCE

Pointer (penunjuk) adalah kemampuan membuat reference ke lokasi data pada memory yang sama, tanpa menduplikasi data yang sudah ada. Sederhananya, dengan kemampuan pointer, kita bisa membuat pass by reference. Dengan contoh kita dengan pointer kini address1 dan address2 menunjuk ke alamat memori yang sama.

- Variabel biasa digunakan untuk menyimpan nilai.
- Variabel pointer digunakan untuk menyimpan alamat dari variabel lainnya.
- Pointer adalah representasi simbolik dari alamat.
- Pointer digunakan untuk mengakses memori dan memanipulasi alamat

Intinya pointer adalah kemampuan membuat reference ke lokasi data di memory yang sama, tanpa menduplikasi data yang sudah ada. Dengan kemampuan pointer kita bisa membuat **pass by reference**. Jadi jika address2 berubah maka juga akan mempengaruhi address1.





Jika menggunakan contoh yang alamat rumah tadi dengan skenario si Ridho punya saudara di rumahnya otomatis saudaranya punya alamat yang sama.

Jenis jenis Operator dalam pointer yaitu:

- Address-of Operator (&)

- Dereference Operator (*)

4. ADDRESS-OF OPERATOR (&)

Address-of Operator (&) adalah operator yang memungkinkan kita untuk mendapatkan/melihat **alamat memori** yang dimiliki oleh variabel tersebut. Cara menggunakannya adalah dengan meletakan tanda (&) di depan identitas saat pemanggilan variabel. Hal itu akan membuat compiler memberikan alamat memori bukan isi/nilai dari memori tersebut.

```
#include <iostream>
using namespace std;
int main () {
```

```
    string nama = "Ridho Lawe";
    cout << &nama << " adalah alamatnya " << nama <<endl;
    return 0;
}
```

Output:

```
→ g++ addressof.cpp ; ./a.exe
0x7ffe1fc1cd40 adalah alamatnya Ridho Lawe
```

5. DEREFERENCE OPERATOR (*) DAN MEMBUAT POINTER

Jika pada address-of operator memungkinkan kita untuk melihat alamat dari variabel yang telah kita buat, maka untuk dereference operator ini berlaku kebalikannya di mana operator ini memungkinkan kita untuk melihat nilai yang tersimpan pada alamat memori.

Sebelum itu kita akan belajar bagaimana cara membuat sebuah variabel pointer

kita sudah tau kalau operator & (ampersand) dipakai untuk mengambil alamat/address jika kita ingin memasukkan address sebagai value ke dalam Variabel maka kita memerlukan operator * (asterisk).

```
// buatPtr.cpp
#include <iostream>
using namespace std;

int main () {
      string var = "Aku Variabel";
      // berikut adalah variabel pointer bernama varPtr
      // yang menampung alamat dari variabel var
      // dengan begini akan ada variabel varPtr yang memiliki value alamat
      // dari variabel var

      string *varPtr = &var; // ini adalah variabel pointer

      // jika kita print varPtr maka akan mengeluarkan output alamat dari
      // variabel var
      cout << "Hasil dari varPtr: " << varPtr << endl;

      // karena varPtr adalah pointer yang menunjuk ke var maka varPtr
      // bisa menggunakan value dari var
```

```
    // dengan ini kita memanfaatkan operator Dereference ( * )

    cout << "Hasil dari *varPtr: " << *varPtr <<endl;

    cout << endl << "Kesimpulannya varPtr isi nya alamatnya var"<<endl;
    cout << "*varPtr hasilnya value dari var"<<endl;
    cout << "Jika masih bingung bisa amati output berikut"<<endl;
    cout <<endl;

    cout << "Hasil/value dari var " << var <<endl;
    cout << "Hasil/value dari alamat var (&var) " << &var<<endl;
    cout << "Hasil/value dari varPtr " << varPtr <<endl;
    cout << "Hasil/value dari *varPtr " << var <<endl;

    return 0;
}
```

Output:

```
→ g++ buatptr.cpp ; ./a.exe
Hasil dari varPtr: 0x7ffc0b3f9250
Hasil dari *varPtr: Aku Variabel


Kesimpulannya varPtr isi nya alamatnya var
*varPtr hasilnya value dari var
Jika masih bingung bisa amati output berikut


Hasil/value dari var Aku Variabel
Hasil/value dari alamat var (&var) 0x7ffc0b3f9250
Hasil/value dari varPtr 0x7ffc0b3f9250
Hasil/value dari *varPtr Aku Variabel
```

6. POINTER PADA ARRAY

Pointer dapat juga digunakan untuk melakukan operasi pada array, di mana pointer akan menunjuk ke alamat memori baik sebagai array keseluruhan maupun untuk tiap nilai yang terdapat pada array. Contoh dapat dilihat pada gambar di bawah.

```cpp
// arrayPtr.cpp
#include <iostream>
using namespace std;

int main () {

      cout << "pointer yang menunjuk ke suatu array"<< endl;
      int a[5] = {1,2,3,4,5};
      int (*aPtr)[5] = &a;
      for (int i =0; i <5; i++){
      cout << *aPtr <<endl;
      }


      cout << "Pointer yang menunjuk ke arah elemen array" <<endl;
      int b[5] = {1,2,3,4,5,};
      int *bPtr = b;
      for (int i = 0; i < 5; i++){
      cout << bPtr <<endl;
      // cout << *bPtr <<endl;
      bPtr++;
      }
      return 0;
}
```

Source code tersebut akan menghasilkan *output* seperti ini:

```
→ g++ arrayPtr.cpp ; ./a.exe
pointer yang menunjuk ke suatu array
0x7fffb701ac10
0x7fffb701ac10
0x7fffb701ac10
0x7fffb701ac10
0x7fffb701ac10
Pointer yang menunjuk ke arah elemen array
0x7fffb701ac30
0x7fffb701ac34
```

```
0x7fffb701ac38
0x7fffb701ac3c
0x7fffb701ac40
```

7. POINTER SEBAGAI PARAMETER FUNGSI

Pointer di function adalah konsep di mana kita menggunakan pointer sebagai parameter atau nilai kembalian (return value) dalam fungsi. Ini memungkinkan kita untuk mengirim alamat memori, menghemat memori, mengembalikan lebih dari satu nilai. Keuntungan menggunakan pointer di function adalah efisiensi memori karena dapat menghindari penyalinan data yang besar seperti struct atau array, dan dapat memodifikasi data asli.

Contohnya tanpa pointer kita punya fungsi untuk mengubah nilai variabel a:

```cpp
// funcNoPtr.cpp
#include <iostream>
using namespace std;

int ubahNilai(int a, int b){
      return a=b;
}
int main () {
      int a,b;
      a=5;
      b=20;
      ubahNilai( a, b );
      cout << a;
      return 0;
}
```

Output:

```
→ g++ funcNoPtr.cpp ; ./a.exe
5⏎
```

Seperti yang terlihat nilai variabel a tidak berubah, karena perubahan hanya terjadi di variabel lokal, variabel a di fungsi adalah duplikasi value dari variabel di main function.

Untuk bisa **mengakses variabel dari variabel aslinya** kita menggunakan **pointer di parameter**. Untuk melakukan ini ada beberapa cara:

a. Penggunaan Address-of Operator (&) sebagai parameter fungsi

```cpp
// funcPtr.cpp
#include <iostream>
using namespace std;

int ubahNilai(int &a, int b){
      return a=b;
}
int main () {
      int a,b;
      a=5;
      b=20;

      ubahNilai( a, b );
      cout << a;

      return 0;
}
```

Output:

```
→ g++ funcPtr.cpp ; ./a.exe
20⏎
```

b. Penggunaan Dereference Operator (*) sebagai parameter fungsi

```cpp
#include <iostream>
using namespace std;

int ubahNilai(int *a, int b){
      return *a=b;
}
int main () {
      int a,b;
      a=5;
      b=20;
```

```
    ubahNilai( &a, b );
    cout << a;

    return 0;
}
```

Output:

```
→ g++ funcPtr.cpp ; ./a.exe
20⏎
```

Kedua cara tersebut memberi akses pada variabel asli di main.

8. POINTER PADA STRUCT

Pointer pada struct adalah konsep di mana kita menggunakan pointer untuk mengakses dan memanipulasi data yang ada di dalam sebuah struct. Struct adalah tipe data kustom yang menggabungkan beberapa variabel dengan tipe data yang berbeda, dan pointer memungkinkan kita untuk bekerja langsung dengan alamat memori dari struct tersebut.

Penggunaan pointer pada struct maupun class juga sangat penting untuk diketahui, karena konsep ini akan sangat berguna untuk memahami bagaimana tiap objek saling terkait, namun pada praktikum Algoritma dan Pemrograman Lanjut ini kita tidak akan membahasnya terlalu dalam. Pembahasan terkait ini akan lebih ditekankan pada praktikum Struktur Data. Namun tidak ada salahnya untuk mengenal bagaimana cara kerja pointer pada struct. *Source code* dapat kita lihat pada gambar di bawah.

```
#include <iostream>
using namespace std;

struct Menu {
    string nama;
    float harga;
};

int main () {
    Menu nasgor;
    Menu *nasgorPtr = &nasgor;

    nasgor.nama = "Nasi Goreng Cumi Hitam Pak Kris";
    nasgor.harga = 15000;
```

```
    cout << nasgor.nama << " " << nasgor.harga<<endl;

    // kita bisa mengakses/manipulasi value variabel nasgor
    // melalui variabel nasgorPtr
    nasgorPtr->nama = "Nasi Goreng Mawut";
    nasgorPtr->harga = 13000;
    cout << nasgor.nama << " " << nasgor.harga <<endl;

    // kalau memberi value pada atribut biasa pakai ( . )
    // kalau memberi value pada atribut dari pointer pakai ( -> )

    return 0;
}
```

Berdasarkan code di atas, dapat dilihat bahwa terdapat perbedaan cara untuk mengakses atribut pada objek struct. Untuk memberi nilai pada atribut objek digunakan Dot Operator (.), namun hal ini dapat berubah apabila kita menggunakan pointer, untuk memberikan nilai pada atribut objek yang ditunjuk oleh pointer, diperlukan operator yang berbeda, yaitu Arrow Operator (->).

STUDI KASUS

1. Buatlah sebuah program dalam bahasa C++ yang menggunakan pointer untuk menukar nilai dua variabel integer. Program harus menampilkan nilai sebelum dan setelah penukaran.
2. Buatlah program dalam bahasa C++ yang menggunakan pointer untuk menghitung panjang sebuah string (tanpa menggunakan fungsi strlen). Program harus menampilkan panjang string yang diinputkan oleh pengguna.