Pointer and Array Review

Pointer

Pointer dalam C adalah sebuah variabel yang menyimpan alamat memori. Dalam konteks data structure, variabel pointer ini akan banyak digunakan untuk membuat berbagai macam data structure seperti linked list, hash table, binary tree, avl, dkk.

Dalam penggunaan pointer, kita biasanya akan banyak menggunakan 2 simbol berikut:

- `*` `(asterisk)`: untuk inisialisasi dan dereferensi pointer
- `&` `(ampersand)`: untuk mengembalikan memori sebuah variabel

Variabel pointer biasanya dideklarasikan menggunakan format:

```
data_type* nama_variabel;

int* pInteger; //deklarasi variabel bernama pInteger yang menunjuk ke datatype integer
char* pChar; //deklarasi variabel bernama pCharacter yang menunjuk ke datatype char
```

Untuk mendeklarasikan sebuah variabel pointer, kita akan menggunakan simbol `*` `(asterisk)` setelah datatype.

Inisialisasi pointer biasanya dilakukan dengan cara:

```c
int num = 10;

int* pNum = &num; //disini pointer akan menunjuk variabel num

//mengecek alamat variabel dan pointer
printf("%d\n", &num);
printf("%d\n", pNum);

//mengecek value variabel dan pointer
printf("%d\n", num); // 10
printf("%d\n", *pNum); // 10
```

Diatas, kita pertama mendeklarasikan variabel bernama `num` dengan value `10` dan juga variabel pointer bernama pNum dengan value address dari variabel num. Address `num` didapatkan dengan menggunakan `&num`.

Untuk melihat address variabel:

- printf dengan parameter `&num` digunakan untuk mendapatkan address dari `num`
- printf dengan parameter `pNum` digunakan untuk mendapatkan value dari `pNum`

Jika operasi diatas dilakukan, karena `pNum` menunjuk variabel `num`, maka address dari `num` dan value dari `pNum` akan sama.

Untuk melihat value variabel:

- printf dengan parameter `num` digunakan untuk mendapatkan value dari `num`.
- printf dengan parameter `*pNum` (dereferensi) digunakan untuk mendapatkan value dari address yang disimpan oleh `pNum`.

Jika operasi diatas dilakukan, maka output dari kedua print akan bernilai sama.

```c
int num = 10;

int* pNum = &num; //disini pointer akan menunjuk variabel num

//mengecek value variabel dan pointer
printf("%d\n", num); // 10
printf("%d\n", *pNum); // 10

num = 12;

//mengecek value variabel dan pointer
printf("%d\n", num); // 12
printf("%d\n", *pNum); // 12
```

Jika variabel yang ditunjuk oleh sebuah pointer diubah valuenya, seperti contohnya `num` yang diubah valuenya dari `10` menjadi `12`, maka value dereferensi pointer juga akan ikut berubah.

```c
int* pNum = NULL; //disini pointer tidak menunjuk ke alamat apapun
```

Kita juga bisa menginisialisasikan sebuah variabel pointer dengan alamat `NULL` untuk menunjukan bahwa pointer tersebut tidak menunjuk ke alamat manapun.

Array

Array dalam C adalah sebuah data structure bawaan yang digunakan untuk menyimpan kumpulan data yang memiliki data type yang sama dalam sebuah memori yang berurutan.

Dalam penggunaan array, kita biasanya akan banyak menggunakan terminologi berikut:

- `index`: untuk mengakses value dari array (mulai dari 0)

`intArray[0]`, intArray indeks ke `0`

`intArray[1]`, intArray indeks ke `1`

`intArray[2]`, intArray indeks ke `2`

Variabel array biasanya dideklarasikan menggunakan format:

```
data_type nama_array[ukuran];

int intArray[10]; //deklarasi variabel integer array bernama intArray dengan ukuran 10
char charArray[20]; //deklarasi variabel chararray bernama charArray dengan ukuran 20
```

Untuk mendeklarasikan sebuah variabel array, kita bisa menggunakan `[x]` setelah nama variabel dimana `x` adalah ukuran dari array.

Array yang berukuran 10 akan dapat menampung 10 value, seperti contohnya array integer berukuran 10 akan dapat menampung 10 value yang memiliki data type integer.

Inisialisasi array biasanya dilakukan dengan cara:

```
int intArray[5] = {1, 2, 3, 4, 5}; // kita memasukkan 5 value sekaligus
int intArray[] = {1, 2, 3, 4, 5}; // tanpa memberi tahu size secara langsung

int intArray[3] = {1, 2, 3, 4, 5}; // akan menghasilkan error


bool boolArr[] = {true, false, true} // bisa

bool boolArr[3];
boolArr[3] = {true, false, true} // tidak bisa


char charArray[3];

// kita memasukan value kedalam array satu per satu
charArray[0] = 'A';
charArray[1] = 'B';
charArray[2] = 'C';

char string[20] = "aku adalah string";

char charArray[]; // akan menghasilkan error
```

Kita dapat memasukkan value kedalam array dengan langsung menggunakan kurung kurawal yang berisikan value `{1, 2, 3, 4, 5}`. Disini kita akan memasukan 5 value kedalam array kita.

Perlu dicatat bahwa kita hanya dapat melakukan teknik ini jika kita mendeklarasikan variabelnya bersamaan dengan inisiasliasi, seperti

`int intArray[5] = {1, 2, 3, 4, 5};`

Jika kita memasukkan value kepada variabel yang sudah sebelumnya dideklarasikan, maka akan menghasilkan error seperti contohnya

`boolArr[3] = {true, false, true};`

Cara kedua untuk memasukkan value kedalam array adalah dengan cara mengakses indeks array satu per satu, seperti contohnya:

```
charArray[0] = 'A';
charArray[1] = 'B';
charArray[2] = 'C';
```

`char` array sendiri special karena kita dapat langsung memasukkan banyak value kedalamnya dengan menggunakan `“ (kutip dua)`. Array dari sebuah `char` biasanya disebut `string`.

Array dapat memiliki dimensi lebih dari satu, array seperti itu dapat dideklarasikan menggunakan format:

```c
int intArray1D[5]; // array ber datatype integer 1 dimensi dengan size 5
int intArray2D[5][5]; // array ber datatype integer 2 dimensi dengan size 5 x 5
int intArray3D[5][5][5]; // array ber datatype integer 3 dimensi dengan size 5 x 5 x 5


// inputting value

int row = 3;
int col = 3;

int intArray1D[row][col];


for(int i = 0; i < row; i++){
  for(int j = 0; j < col; j++){
    scanf("%d ", &intArray1D[i][j]);
  }
}

// printing value

int row = 3;
int col = 3;

int intArray1D[row][col] = {{0, 1, 2},
                            {3, 4, 5},
                            {6, 7, 8}}

for(int i = 0; i < row; i++){
  for(int j = 0; j < col; j++){
    printf("%d ", intArray1D[i][j]);
  }
  printf("\n")
}

// output:
// 0 1 2
// 3 4 5
// 6 7 8
```

Cara memasukkan value ke dalam array multidimensi juga sama seperti array satu dimensi, operasi read dan edit juga sama.

Struct

Struct dalam C adalah konstruksi yang dapat membuat kita menggabungkan beberapa tipe data menjadi satu kelompok.

Struct dapat didefinisikan menggunakan fomat:

```
struct NamaStruct {
  data_type1 nama_variabel1;
  data_type2 nama_variabel2;
  ...
};

struct People {
  int age;
  char name[100];
  float height;
};
```

Struct sendiri bisa diisi dengan berapa banyak pun attribut (data type).

Di contoh diatas, `struct People` memiliki tiga atribut, yaitu:

- `age` yang memiliki data type `int`
- `name` yang memiliki data type `char array`
- `height` yang memiliki data type `float`.

Memasukkan data kedalam struct dapat dilakukan dengan cara berikut:

```c
#include <stdio.h>
#include <string.h>
struct People {
  int age;
  char name[100];
  float height;
};


int main(){


  People people1 = {17, "Sen", 170};
  People people2;

  people2.age = 20;
  strcpy(people2.name, "Alvin");
  people2.height = 180;

  printf("%d %s %d", people2.age, people2.name, people2.height);

  return 0;
}
```

Dicontoh diatas, kita bisa pertama mendeklarasikan struct dengan cara

`nama_struct nama_variabel;`

Disini kita bisa langsung memasukkan value kedalam struct tersebut dengan langsung menggunakan kurung kurawal yang berisikan value-value dalam struct tersebut seperti `{17, "test", 190}`. Note cara ini hanya dapat dilakukan saat kita mendeklarasikan structnya sekaligus.

Selain itu kita juga bisa memasukkan value kedalam struct dengan cara mengakses `nama_variabel.attribut = value`

dimana dicontoh diatas kita memasukkan attribut

`age` dengan value `20`

`name` dengan value `Alvin`

`height` dengan value `180`

Kita juga dapat membuat array of structs yang dapat menyimpan banyak data.

```c
#include<stdio.h>

struct People {
  int age;
  char name[100];
  float height;
};

int main(){

  int amount = 10;
  People people[amount];

  for(int i = 0; i < amount; i++){
    scanf("%d %s %d", &people[i].age, &people[i].name, &people[i].height);
  }

  for(int i = 0; i < amount; i++){
    printf("%d %s %d\n", people[i].age, people[i].name, people[i].height);
  }

  return 0;
}
```

Disini kita dapat menggunakan for loop untuk memasukkan value-value kedalam struct `People` kita. Kita juga dapat menggunakan for loop untuk memprint value-value yang telah kita masukkan tadi.