COMPUTER ENGINEERING

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

Praktikum
Dasar Pemrograman

Perulangan Percabangan Array String

2024

1 Tujuan

- Mahasiswa mengenal dan mampu menggunakan ekspresi-ekspresi logika dan perbandingan pada bahasa pemrograman C
- Mahasiswa mengenal dan mampu menggunakan syntax-syntax percabangan pada bahasa pemrograman C
- Mahasiswa dapat mengenal dan menggunakan perulangan while pada bahasa C
- Mahasiswa dapat mengenal dan menggunakan perulangan do-while pada bahasa C
- Mahasiswa dapat mengenal dan menggunakan perulangan for pada bahasa C
- Mahasiswa dapat mengenal dan menggunakan array dimensi satu maupun multidimensi.
- Mahasiswa mampu memanfaatkan perulangan untuk mengolah data pada array.
- Mahasiswa dapat mengenal dan menggunakan string.

2 Ekspresi Logika dan Perbandingan

2.1 Ekspresi Perbandingan

Berikut adalah operator-operator yang digunakan pada suatu ekspresi perbandingan

Tabel 1: Operator Perbandingan

Operator	Nama	Contoh Ekspres
==	Sama Dengan	x == y
!=	Tidak Sama Dengan	x != y
>	Lebih Dari	x >y
<	Kurang Dari	x <y
>=	Lebih Dari Sama Dengan	x >= y
<=	Kurang Dari Sama Dengan	x <= y

Suatu ekspresi perbandingan akan mengembalikan nilai berupa `true` atau `false` yang ditandai dengan nilai 0 atau 1. Sebagai contoh:

```
printf("%d",0>1); // Akan Mencetak 0 ke layar
printf("%d",0<1); // Akan Mencetak 1 ke layar
```

2.2 Ekspresi Logika

Berikut adalah operator-operator logika yang digunakan pada suatu ekspresi logika

Tabel 2: Ekspresi Logika

Operator	Nama	Contoh Ekspresi
&&	AND	$x < 5 \;\&\&\; x < 10$
||	OR	$x < 5 \;\|\|\; x < 4$
!	NOT	$!(x < 5 \&\& x < 10)$

Sama seperti ekspresi perbandingan, ekspresi logika akan mengembalikan nilai berupa true atau false

3 Percabangan

3.1 Pernyataan If

Pernyataan `if` digunakan untuk menentukan blok kode C yang dijalankan apabila ekspresi kondisi bernilai benar (TRUE),

```
// Block of code before if
if (Condition)
{
 // Blok kode yang akan dieksekusi jika kondisinya benar(True).
}
// Blok kode setelah if
```

Sebagai contoh, perhatikan program berikut

Listing 1: Contoh Pernyataan If

```
1  include <stdio.h>
2
3  int main()
4  {
5    //Deklarasi variabel
6    int uangSaya,hargaRoti;
7    uangSaya = 5000;
8    hargaRoti = 10000;
9
10   if (uangSaya>=hargaRoti)
11   {
12       printf("saya bisa beli roti\n");
13   }
14   printf("hehe");
15   return 0;
16 }
```

Keluaran program ini

```
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=10000` maka output dari program ini akan menjadi

```
saya bisa beli roti
hehe
```

3.2 Pernyataan If-else

Pernyataan `else` digunakan untuk menentukan blok kode yang di jalankan apabila kondisi salah.

```
// Blok kode sebelum if
if (Condition)
{
 // Blok kode yang akan dieksekusi jika kondisinya benar
} else
```

```
{
 // Blok kode yang akan dieksekusi jika kondisinya salah
}
// Blok kode setelah pernyataan if-else
```

Berikut contoh penggunaan if-else

Listing 2: if-else example

```
1  include <stdio.h>
2
3  int main()
4  {
5    //Deklarasi variabel
6    int uangSaya,hargaRoti;
7    uangSaya = 5000;
8    hargaRoti = 10000;
9
10   if (uangSaya>=hargaRoti)
11   {
12       printf("saya bisa beli roti\n");
13   }
14   else
15   {
16         printf("saya tidak bisa beli roti\n");
17   }
18   printf("hehe");
19   return 0;
20 }
```

Keluaran program adalah sebagai berikut

```
saya tidak bisa beli roti
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=10000` maka output dari program ini akan menjadi

```
saya bisa beli roti
hehe
```

3.3 Pernyataan if-else if

pernyataan `else if` digunakan untuk menjalankan blok kode apabila kondisi statement `if` atau `else if` sebelumnya bernilai salah.

```
// blok kode sebelum if
   if (Condition1)
   {
  /* blok kode yang akan dieksekusi jika Kondisi 1
  adalah benar*/
   }
   else if (Condition2)
   {
```

```
  /* blok kode yang akan dieksekusi jika Kondisi 1 salah
  dan Kondisi 2 benar */
   }
   else if (Condition3)
   {
  /* Blok kode yang akan dieksekusi kapan
  Kondisi 1 dan Kondisi 2 salah dan
  Kondisi 3 benar*/
   }
   ...
   else if (ConditionN)
   {
  /*Blok kode yang akan dieksekusi kapan
  Kondisi 1 hingga KondisiN-1 salah dan
  Kondisi N benar*/
   }
   else
   {
  /* Blok kode yang akan dieksekusi kapan
  Kondisi 1 hingga Kondisi N salah*/
   }
// Blok kode setelah if
```

Berikut contoh penggunaan if-else if

Listing 3: Contoh if-else if

```
1  include <stdio.h>
2
3  int main()
4  {
5    //Deklarasi variabel
6    int uangSaya,hargaRoti;
7    uangSaya = 5000;
8    hargaRoti = 10000;
9
10   if (uangSaya>hargaRoti)
11   {
12       printf("saya bisa beli roti\n");
13   }
14   else if(uangSaya==hargaRoti)
15   {
16       printf("saya bisa beli roti tapi uang saya akan langsung habis\n");
17   }
18   else
19   {
20         printf("saya tidak bisa beli roti\n");
21   }
22   printf("hehe");
23   return 0;
24 }
```

Output dari program ini adalah

```
saya tidak bisa beli roti
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=10000` maka output dari program ini akan menjadi

```
saya bisa beli roti tapi uang saya akan langsung habis
hehe
```

Jika baris ke 7 diganti dengan `uangSaya=12000` maka output dari program ini akan menjadi

```
saya bisa beli roti
hehe
```

3.4 Nested if

Nested if merupakan konsep di mana di dalam suatu blok if terdapat pernyataan if.

```
// Blok kode sebelum if
if (Condition1)
{
    if (Condition2)
    {
        // lakukan sesuatu
    }
    else
    {
        // lakukan sesuatu yang lain
    }
}
else
{
    // melakukan sesuatu yang lain
}
```

Berikut contoh penggunaan nested if

Listing 4: Contoh nested if

```
1  include <stdio.h>
2
3  int main()
4  {
5    // Declare the variables
6    int myMoney,breadPrice,friendsMoney;
7    myMoney = 5000;
8    breadPrice = 10000;
9    friendsMoney = 42069;
10
11
12   if (myMoney>breadPrice)
```

```
13      {
14          printf("I can buy bread\n");
15      }
16      else if(myMoney==breadPrice)
17      {
18          printf("I can buy bread but I will ran out of money\n");
19      }
20      else
21      {
22          if(friendsMoney+myMoney >= breadPrice)
23          {
24              printf("I can buy bread if I borrow my friend money\n");
25          }
26          else
27          {
28                printf("I can't buy bread\n");
29          }
30      }
31      printf("hehe");
32      return 0;
33  }
```

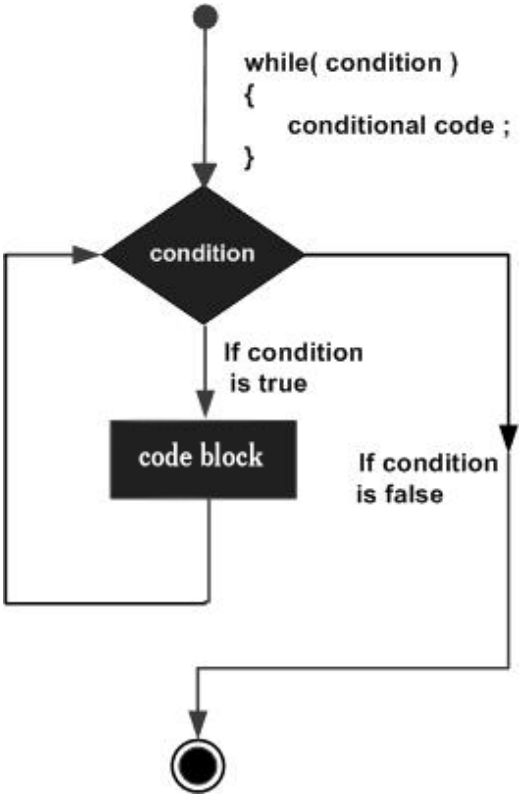
3.5 Tugas Pendahuluan

1. Apa tujuan dari percabangan dalam pemrograman?
2. Selain menggunakan statement if, percabangan juga bisa dilakukan dengan statement switch-case. Jelaskan apa saja yang kamu ketahui tentang switch-case!
3. Buatlah program yang menerima input 3 buah bilangan bulat A, B, dan C. Outputkanlah 3 bilangan bulat itu ke layar dengan urutan paling besar ke paling kecil. Lakukanlah ini dengan menggunakan statement if, if else, if else if, atau nested if.

4 Perulangan

4.1 Perulangan while

Perulangan while akan menjalankan blok kode yang berada di dalamnya selama kondisi perulangan masih bernilai benar.



Gambar 1: Flow chart perulangan while

Syntaxnya pada bahasa C adalah sebagai berikut:

```
while(Condition)
{
    // Blok kode yang akan diulang
}
```

Sebagai contoh, perhatikan kode berikut

Listing 5: Contoh Penggunaan while

```
1  int main()
2  {
3    int uangSaya,hargaRoti;
4    uangSaya = 10000;
5    hargaRoti = 2000;
6    while(uangSaya >= hargaRoti)
7    {
8        printf("Beli roti 1, uang saya sisa %d", uangSaya - hargaRoti);
9        uangSaya -= hargaRoti;
10   }
11   printf("Uang saya tidak cukup lagi");
12   return 0;
13 }
```

Keluaran program di Listing 5 adalah sebagai berikut

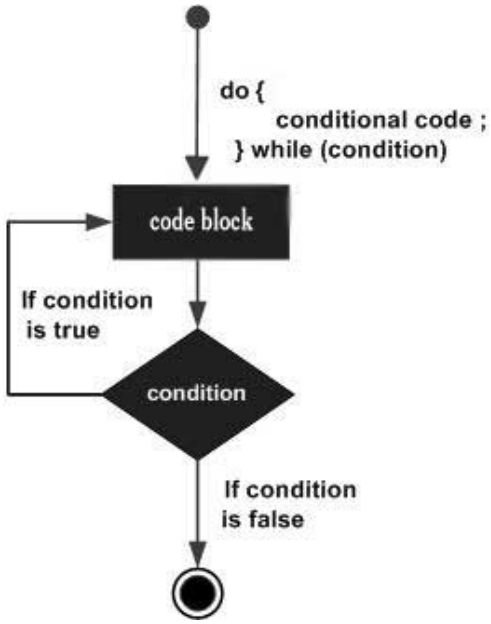
```
Beli roti 1, uang saya sisa 8000
Beli roti 1, uang saya sisa 6000
Beli roti 1, uang saya sisa 4000
```

```
Beli roti 1, uang saya sisa 2000
Beli roti 1, uang saya sisa 0
Uang saya tidak cukup lagi
```

Pada contoh ini, operasi pada baris 9 membuat variabel `uangSaya` berkurang 2000 pada setiap pengulangan hingga akhirnya nilai `uangSaya` tidak lebih dari atau sama dengan `hargaRoti` lagi. Kondisi perulangan akan menjadi tidak valid dan akhirnya keluar dari perulangan. Kemudian ia mencetak "Uang saya tidak cukup lagi", perintah setelah pernyataan while loop.

4.2 do-while loop

do-while loop sebenarnya sama seperti while loop hanya saja do-while akan menjalankan perintah pada blok kode didalamnya sekali sebelum melakukan pengecekan kondisi.



Gambar 2: Pernyataan do-while

Syntaxnya pada bahasa C adalah sebagai berikut:

```
do{
    // the block of code that will be repeated
}while(Condition)
```

Sebagai contoh, perhatikan kode berikut

Listing 6: Contoh Penggunaan do-while

```
1 int main()
2 {
3   int uangSaya,hargaRoti;
4   uangSaya = 10000;
5   hargaRoti = 12000;
6   do{
7       printf("Beli roti 1, uang saya sisa %d", uangSaya - hargaRoti);
8       uangSaya -= hargaRoti;
9   }while(uangSaya >= hargaRoti)
```

```
10    printf("Uang saya tidak cukup lagi");
11    return 0;
12  }
```

Output dari program pada Listing 6 adalah

```
Beli roti 1, uang saya sisa -2000
Uang saya tidak cukup lagi
```

Variabel `uangSaya` dikurangi dengan `hargaRoti` sebelum memeriksa `uangSaya>=hargaRoti` kondisi. Seandainya kode di atas menggunakan perulangan while, blok kode yang berulang tidak akan diek-sekusi sekali pun.

4.3 Perulangan for

Misalkan terdapat blok kode while dengan bentuk seperti ini:

```
InitializationStatement; // e.g.: int i = 0;
while(Condition){
    // do something
    updateStatement; // e.g.: i++
}
```

Hal ini setara dengan

```
for(InitializationStatement;Condition;updateStatement){
    // do something
}
```

Sebagai contoh, perhatikan program berikut:

Listing 7: Contoh Penggunaan for

```
1  int main()
2  {
3      int i=0;
4      for(i=1;i<10;i++){
5          printf("%d ",i);
6      }
7    return 0;
8  }
```

Output dari program ini adalah

```
1 2 3 4 5 6 7 8 9
```

Berikut kode pada Listing 7 jika diubah menjadi bentuk while-loop

Listing 8: For dalam bentuk while

```
1  int main()
2  {
3      int i=0;
4      i=1;
5      while(i<10){
```

```
6           printf("%d ",i);
7           i++;
8       }
9   return 0;
10 }
```

Catatan: Terdapat keyword break dan continue yang bisa digunakan untuk mengendalikan (kontrol) alur pada perulangan. Pelajari secara mandiri!

4.4 Tugas Pendahuluan

1. Apa yang terjadi jika kita menuliskan `break;` dalam blok kode sebuah perulangan?
2. Buatlah program dalam bahasa C yang menghitung faktorial dari sebuah bilangan bulat non-negatif yang dimasukkan oleh pengguna menggunakan loop do-while. Tampilkan hasilnya.
3. Buatlah program dalam bahasa C untuk mencari bilangan prima antara 1 dan 100. Gunakan loop for untuk mengiterasi melalui semua angka dan pernyataan continue untuk mengabaikan angka yang bukan prima. Tampilkan semua bilangan prima yang ditemukan.

5 Array

Array adalah koleksi data dimana setiap elemen mempunyai nama yang sama dan bertipe sama. Setiap elemen diakses berdasarkan indeks elemennya.

5.1 Array 1D

Variabel array dimensi satu dideklarasikan dengan menentukan jenis elemen dan jumlah elemen yang di perlukan oleh array.

Syntax:

```
DataType variableName [arraySize];
```

1. `DataType`.
   Tipe data yang akan digunakan untuk array :`float`,`int`,`char` dsb
2. `variableName`
   Nama variabel mengikuti aturan pemberian nama variabel,
3. `arraySize`
   konstanta integer lebih besar dari 0.

Untuk menginisialisasi array dimensi satu, dapat dilakukan dengan cara seperti berikut:

```
int contoh_array[5] = {4,2,0,6,9};
```

Data di dalam array dapat akses dengan menggunakan suatu bilangan yang merupakan index dari array tersebut. Perhatikan potongan kode berikut.

Listing 9: Contoh Mengakses Array 1D

```
int main()
{
    int arr[5] = {4,2,0,6,9};
    printf("%d\n",arr[0]);
    printf("%d\n",arr[4]);
    int i = 0;
    printf("%d\n",arr[i]);
    for(i=0;i<5;i++)
        printf("%d",arr[i]);
}
```

Potongan kode pada Listing 10 akan memberikan output

```
4
9
4
42069
```

5.2 Array 2D dan Array Multidimensi lainnya

Array dimensi dua pada dasarnya hanya merupakan array dimensi satu di dalam array dimensi satu. Oleh karena itu, untuk mendeklarasikan array dimensi dua kita dapat menggunakan syntax seperti berikut.

```
DataType variableName[arraySize1][arraySize2];
```

Hal ini berlaku juga untuk array dengan dimensi lebih dari dua.

```
DataType variableName[arraySize1]...[arraySizeN];
```

Akan ada $arraySize_1 \times arraySize_2 \times \cdots \times arraySize_n$ elemen yang akan dialokasikan ke memori setelah melakukan array multidimensi seperti itu

Untuk menginisialisasi suatu array multidimensi dapat dilakukan sama seperti array biasa:

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

5.3 Tugas Pendahuluan

1. Buatlah suatu program yang menerima input dari pengguna berupa angka 1 hingga 9, lalu memasukkan semua angka tersebut ke dalam suatu array!
2. Apakah yang akan terjadi jika suatu array `arr` diakses dengan `arr[-1]`?
3. Apakah yang akan terjadi jika suatu array `arr` dengan ukuran 5 diakses dengan `arr[5]`?
4. Lihatlah kode berikut

```
for(i=0;i<10;i++){
    for(j=i;j<10;j++){
        printf("A");
    }
```

```
}
```

Ada berapa banyak huruf A yang akan muncul pada layar jika program tersebut dijalankan?

6 String

Secara umum, string merupakan kumpulan dari satu atau lebih karakter. Spesifik pada bahasa C, string didefinisikan sebagai kumpulan karakter yang diakhiri oleh karakter null `'\0'`.
Misalkan string `"Dasar"`, pada bahasa C direpresentasikan sebagai kumpulan karakter `'D'`, `'a'`, `'s'`, `'a'`, `'r'`, dan `'\0'`.

6.1 Penggunaan String

Karena string tidak lain adalah array dari char, maka cara pembuatan tipe data string dalam bahasa C juga sama seperti cara pembuatan array. Berikut contohnya:

Listing 10: Contoh String dari Char

```c
#include <stdio.h>

int main(void)
{
char foo[8] = {'b','e','l','a','j','a','r','\0'};
printf("Isi variabel foo adalah %s \n", foo);

return 0;
}
```

`'\0'` adalah salah satu syarat pembuatan string di dalam bahasa C. Semua string harus memiliki karakter "khusus" untuk menandakan akhir dari string. Tanda `'\0'` mewakili karakter null yang dipakai oleh compiler bahasa C sebagai tanda akhir sebuah string.

Contoh source code penggunaan `scanf` untuk membaca string:

Listing 11: Contoh String dengan scanf

```c
#include <stdio.h>

int main() {
  // Mendeklarasikan variabel untuk menyimpan input dari pengguna
  int age;
  float height;
  char name[50];

  // Meminta pengguna untuk memasukkan usia mereka
  printf("Masukkan usia Anda: ");
  scanf("%d", &age);

  // Meminta pengguna untuk memasukkan tinggi mereka
  printf("Masukkan tinggi Anda (dalam meter): ");
  scanf("%f", &height);

```

```
17      // Meminta pengguna untuk memasukkan nama mereka
18      printf("Masukkan nama Anda: ");
19      scanf("%s", name);
20
21      // Menampilkan informasi yang dimasukkan pengguna
22      printf("Nama: %s\n", name);
23      printf("Usia: %d tahun\n", age);
24      printf("Tinggi: %.2f meter\n", height);
25
26      return 0;
27    }
```

Contoh source code penggunaan `gets` untuk membaca string:

Listing 12: Contoh String dengan gets

```
1  #include <stdio.h>
2
3  int main () {
4
5    char arr[100];
6    while(true)
7    {
8      gets(arr);
9
10     printf("-- %s\n", arr);
11   }
12   return 0;
13
14 }
```

String yang dibaca dengan mengunakan scanf atau gets akan secara otomatis memiliki `null` character di akhir.

6.2 Fungsi-Fungsi String

Dalam bahasa pemrograman C, terdapat library yang dibuat dengan tujuan memudahkan pengguna dalam mengolah string. Library tersebut tersimpan dalam `<string.h>`, oleh karena itu, untuk mengakses library ini, diperlukan tambahan preprocessor, yaitu:

```
1    #include <string.h>
```

Pelajari berbagai fungsinya di www.cplusplus.com.

6.3 Tugas Pendahuluan

1. Buatlah program dalam bahasa C yang mengambil dua string dari pengguna dan menentukan apakah kedua string tersebut anagram (mengandung karakter yang sama dalam urutan yang berbeda). Misalnya kata "usap" dan "sapu". Tampilkan pesan yang sesuai.

2. Sebutkan 5 fungsi yang terdapat pada library `string.h`! jelaskan kegunaan masing-masing fungsi tersebut!

3. Untuk mengeluarkan output string, selain menggunakan `printf()` kita juga bisa menggunakan `puts()`. Jelaskan kelebihan menggunakan `puts()` jika dibandingkan dengan `printf()`!