

MATERI DAN PRAKTIKUM "PERULANGAN, ARRAY, DAN STRING"

MATA KULIAH: MATRIKULASI STRUKTUR DATA GASAL 2019/2020

Dosen: Dr. Umi Laili Yuhana, S.Kom, M.Sc.

S2 TEKNIK INFORMATIKA ITS

23 OKTOBER 2019

PETUNJUK:

- Pelajari baik-baik materi dalam modul ini
- Buat program dari setiap latihan yang diberikan DAN TULIS NAMA DAN NRP ANDA DALAM SETIAP LATIHAN
- Sajikan hasil program/latihan anda dalam bentuk laporan yang berisi capture dan pembahasan
- Cetak laporan dan kumpulkan di loker paling lambat Hari Kamis, 24 Oktober 2019 Pukul 07.30

=====

Tujuan Sesi Praktikum :

1. Praktikan memahami struktur perulangan dan juga variasinya. Mahasiswa dapat menghindari error yang sering terjadi dalam bahasa C.
2. Praktikan memahami dan dapat mengimplementasikan konsep array dan string.

MATERI & PRAKTIKUM

a. Error yang sering terjadi

Pada pemrograman dengan menggunakan bahasa C, khususnya programmer yang baru belajar bahasa C sering mengalami yang namanya *common error*. Hal ini disebabkan kurang pahamnya atau juga kurang telitinya programmer saat membuat baris-baris codenya.

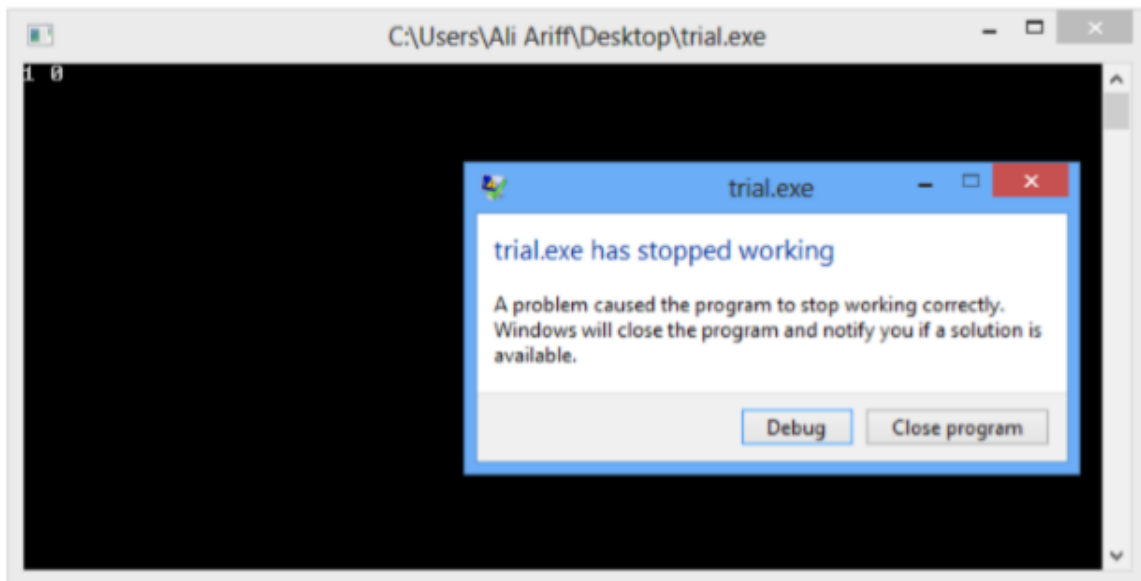
1. Divide by zero (Pembagian dengan nol)

Error ini dapat terjadi jika ada bilangan yang dibagi dengan 0.

Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a,b;
6      scanf("%d%d",&a,&b);
7      printf("%d",a/b);
8      return 0;
9  }
10
```

Hasil tampilan pada layar adalah sebagai berikut:

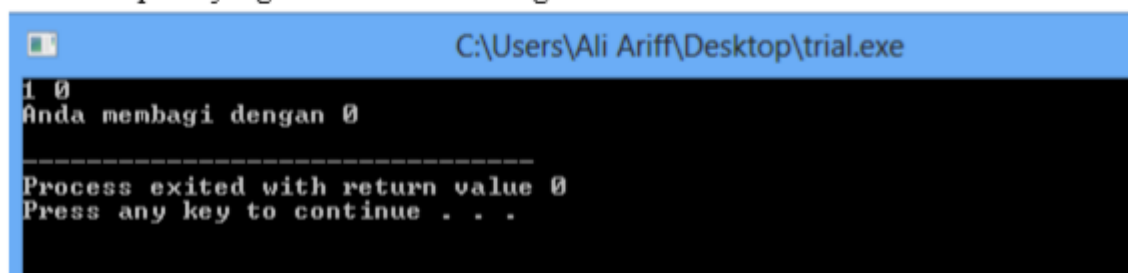


Jika dimasukkan angka 1 0, maka akan terjadi error karena pembagian dengan 0. Untuk mengatasinya kita bisa lakukan pengecekan dulu apakah b sama dengan 0.

Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a,b;
6      scanf("%d%d",&a,&b);
7      if(b!=0)printf("%d",a/b);
8      else printf("Anda membagi dengan 0\n");
9      return 0;
10 }
11
```

Maka tampilan yang muncul adalah sebagai berikut.



2. Integer/Integer

Yang dimaksud integer/integer disini ialah jika kalian melakukan pembagian integer dengan integer hasilnya akan berupa integer juga, banyak programmer pemula di C, menganggap bahwa jika integer/integer lalu dimasukkan pada variabel float/double hasilnya akan berubah menjadi float/double. Hal tersebut tidak akan terjadi pada bahasa C.

Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a,b;
6      double c;
7      scanf("%d%d",&a,&b);
8      if(b!=0)
9      {
10         c = a/b;
11         printf("%lf",c);
12     }
13     else printf("Anda membagi dengan 0\n");
14     return 0;
15 }
16
```

Keluaran dari program tersebut sebagai berikut:

```
5 2
2.000000
-----
Process exited with return value 0
Press any key to continue . . .
```

Perhatikan bahwa 5 dibagi dengan 2 hasilnya bukan 2.5 walaupun dimasukkan ke variabel c yang bertipe double. Untuk mengatasi hal semacam ini salah satunya bisa dengan melakukan *type cast* alias merubah tipe data saat pembagian.

Contoh:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a,b;
6      double c;
7      scanf("%d%d",&a,&b);
8      if(b!=0)
9      {
10         c = (double)a/b;
11         printf("%lf",c);
12     }
13     else printf("Anda membagi dengan 0\n");
14     return 0;
15 }
16
```

Hasilnya:

```
5 2
2.500000
-----
Process exited with return value 0
Press any key to continue . . .
```

Program diatas melakukan casting variable a menjadi double pada saat pembagian, hal ini membuat pembagian $5/2$ menjadi benar 2.5. Type cast dapat dilakukan pada salah satu variabel maupun kedua variabel.

3. Overflow

Overflow adalah keadaan saat sebuah variabel kelebihan kapasitas dari yang seharusnya.

contohnya pada variabel bertipe integer yang memiliki range antara -2.147.483.648 sampai 2.147.483.647 dimasukkan nilai yang diluar dari range tersebut maka akan terjadi overflow yaitu nilainya akan kembali mengulang karena bentuk dari range ini circular.

```
#include <stdio.h>

int main()
{
    int i = 2147483647;
    printf("%d",i);
    return 0;
}
```

Hasilnya:

```
2147483647
-----
Process exited with return value 0
Press any key to continue . . .
```

Sekarang coba tambahkan nilai i dengan 1

```
#include <stdio.h>

int main()
{
    int i = 2147483648;
    printf("%d",i);
    return 0;
}
```

Hasilnya:

```
-2147483648
Process exited with return value 0
Press any key to continue . . .
```

Hasilnya minus karena variabel *i* overflow jadi nilainya akan kembali mengulang dari awal.

4. Infinite Loop

Infinite loop adalah suatu keadaan dimana sebuah loop tidak akan berhenti sampai kapanpun. Ini biasanya disebabkan oleh kesalahan logika programmer saat menulis code.

Contoh:

```
#include <stdio.h>

int main()
{
    int i;
    i = 0;
    while(i<5)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

Hasil dari code diatas akan mencetak angka 0 yang tidak ada habisnya. Hal ini disebabkan oleh kesalahan logika ataupun lupa dalam penulisan perubahan kondisi *i* agar membuat kondisi pengecekan while bernilai false dan akhirnya berhenti.

Kesimpulan dari materi *common error* ini adalah bahwasannya error-error yang diterangkan diatas semua berdasarkan pengalaman saat membuat program. Jadi materi diatas hanya memberikan gambar umum saja. Untuk lebih memahaminya adalah dengan sering-sering membuat program dan nantinya akan mengalami sendiri hal-hal seperti diatas. Untuk itu diharapkan praktikan lebih sering lagi mencoba dan mengeksplorasi lebih dalam materi-materi yang disampaikan oleh dosen

b. Konsep dasar perulangan dan variasinya

Perulangan digunakan untuk mengulangi satu atau lebih perintah tertentu yang dikehendaki programmer guna menyelesaikan masalah tertentu. Dengan menggunakan perulangan, kode program yang dibuat programmer dapat menjadi lebih singkat, karena beberapa perintah yang sama tidak perlu ditulis berulang-ulang. Perulangan juga digunakan untuk mempermudah pemrograman yang berhubungan dengan algoritma yang berpola.

1. Perulangan dengan while

Bentuk umum atau kerangka dari blok while sebagai berikut.

```
<inisialisasi>;  
while (<kondisi>)  
{  
    <pernyataan yang akan dijalankan>;  
    <perubahan kondisi>;  
}
```

Saat memasuki statemen while, program akan mengecek kondisi yang dituliskan setelah statemen while. Jika memenuhi, program akan menjalankan statemen-statemen yang terdapat dalam blok tersebut hingga kondisi yang dituliskan tidak terpenuhi.

Contoh:

```
1  #include <stdio.h>  
2  
3  int main()  
4  {  
5      int i;  
6      i = 0;  
7      while( i < 5 )  
8      {  
9          printf("%d\n", i+1);  
10         i++;  
11     }  
12     return 0;  
13 }
```

Hasil eksekusi :

```
1  
2  
3  
4  
5  
-----  
Process exited with return value 0  
Press any key to continue . . .
```

2. Perulangan dengan do-while

Perulangan akan dilakukan minimal satu kali terlebih dahulu, kemudian baru dilakukan pengecekan terhadap kondisi. Jika kondisi benar maka perulangan masih akan tetap dilakukan.

Bentuk umum do-while

```
<inisialisasi>;  
do  
{  
    <pernyataan yang akan dijalankan>;  
    <perubahan kondisi>;  
}  
while (<kondisi>;
```

Contoh:

```
1  #include <stdio.h>  
2  
3  int main()  
4  {  
5      int i;  
6      i = 0;  
7      do  
8      {  
9          printf("%d\n", i+1);  
10         i++;  
11     }  
12     while( i < 5 );  
13     return 0;  
14 }
```

Hasil eksekusi:

```
1  
2  
3  
4  
5  
-----  
Process exited with return value 0  
Press any key to continue . . .
```

3. Perulangan dengan for

Bentuk umum

```
for (<nilai awal> ; <kondisi> ; <penambahan/penurunan>)  
{  
    <statement-statement yang akan dijalankan>;  
}
```

Contoh:

```
#include <stdio.h>  
  
int main()  
{  
    int i;  
    for(i=0 ; i<5 ; i++)  
    {  
        printf("%d\n", i+1);  
    }  
    return 0;  
}
```

Hasil:

```
1
2
3
4
5
-----
Process exited with return value 0
Press any key to continue . . .
```

Konsep Dasar Array

Adalah kumpulan data bertipe sama yang menggunakan nama sama. Dengan menggunakan array, sejumlah variabel dapat memakai nama yang sama. Antara satu variabel dengan variabel lain di dalam array dibedakan berdasarkan indexnya.

a. Array Satu Dimensi

Merupakan kumpulan data yang bernama sama dengan pembeda sebuah indeks. Contoh penggunaan array satu dimensi:

```
#include <stdio.h>

int main()
{
    // Deklarasi Array
    // indeks dari sebuah array dimulai dari 0 hingga kapasitas-1

    // int a[10] berarti ada array of integer, berkapasitas 10 integer
    // isi dari a adalah random
    int a[10];

    // isi dari b adalah kosong/null
    int b[10] = {};

    // assignment nilai pada array a
    a[0] = 100;
    a[1] = 20;

    // mencetak nilai dalam sebuah array
    printf("%d %d\n", a[0], a[1]);

    return 0;
}
```

Array pertama kali dideklarasikan seperti halnya variabel biasa, tetapi dengan kapasitas di dalam kurung siku, isi array yang bisa diakses adalah antara 0 sampai kapasitas-1, jadi jika mendeklarasikan array berkapasitas 10, indeks yang bisa diakses hanya indeks 0 sampai 9. Kemudian assign dan akses isi array sama dengan variabel biasa, tetapi menggunakan indeks.

Mengapa perlu array? Andaikan kita membutuhkan 1000 inputan data, kita tidak perlu membuat deklarasi variabel a1 hingga a1000, kita cukup menuliskan 1 variabel a dengan tipe array berkapasitas 1000.

Contoh program tanpa array:

```
/*
    Program menginputkan 5 bilangan dan mencetaknya kembali
*/
#include <stdio.h>

int main()
{
    // Deklarasi 5 buah variabel integer
    int a, b, c, d, e;

    // Input 5 buah bilangan kedalam variabel
    scanf("%d %d %d %d %d",&a, &b, &c, &d, &e);

    // Outputkan bilangan yang telah diinputkan
    printf("Bilangan ke-1 adalah %d\n",a);
    printf("Bilangan ke-2 adalah %d\n",b);
    printf("Bilangan ke-3 adalah %d\n",c);
    printf("Bilangan ke-4 adalah %d\n",d);
    printf("Bilangan ke-5 adalah %d\n",e);

    return 0;
}
```

Bayangkan kita tidak hanya memasukkan 5 data, melainkan 1000 data, bagaimana kita mendeklarasikan semuanya dalam variable dan kemudian mencetaknya? Maka dari itulah array digunakan.

Contoh program menggunakan array:

```
/*
    Program menginputkan 5 bilangan dan mencetaknya kembali
*/
#include <stdio.h>

int main()
{
    // Deklarasi 5 buah variabel integer dalam array
    int a[5], i;

    // Input 5 buah bilangan kedalam variabel dengan bantuan for
    for(i = 0; i < 5; i++) scanf("%d",&a[i]);

    // Outputkan bilangan yang telah diinputkan
    for(i = 0; i < 5; i++) printf("Bilangan ke-%d adalah %d\n",i+1,a[i]);

    return 0;
}
```

b. Array Multidimensi

Array Multidimensi adalah array yang dapat menampung lebih dari satu array. Apabila array satu dimensi hanya memiliki sebuah index, array multidimensi memiliki dua atau lebih index untuk mengakses seluruh elemen dalam array tersebut. Contoh program dengan array dua dimensi:

```

#include <stdio.h>

int main()
{
    // Deklarasi Array Multidimensi
    // indeks dimulai dari 0 hingga kapasitas-1 setiap dimensinya

    // int matriks[10][10] berarti ada array of array of integer
    // dengan kapasitas 10x10 integer (100 integer)
    int matriks[10][10];

    // assignment nilai pada array multidimensi
    matriks[2][3] = 100;

    // mencetak nilai dalam sebuah array multidimensi
    printf("%d\n",matriks[2][3]);

    return 0;
}

```

Apabila program diatas dibuat dengan menggunakan array satu dimensi, maka setiap baris pada matriks tersebut harus dideklarasikan sebagai variabel yang berbeda (misal: matriks1[100], matriks2[100], matriks3[100], dst). Tentunya hal ini akan sangat sulit diimplementasikan ditambah lagi apabila baris pada matriks ditentukan oleh user yang menggunakan program dan jumlah baris tersebut sangat besar.

Konsep Dasar String

String adalah tipe data yang menyimpan kumpulan karakter/symbol. Dalam bahasa pemrograman C, suatu string dideklarasikan sebagai *array yang bertipe char*.

Contoh pendeklarasian string (1):

```

#include <stdio.h>

int main()
{
    // Deklarasi String (1)
    char array[] = "Halo";

    return 0;
}

```

Contoh di atas akan mendeklarasikan string bernama array dengan kapasitas 5 karakter, di mana array[0] = „H“, array[1] = „a“, array[2] = „l“, array[3] = „o“, dan array[4] = „\0“.

Perhatikan bahwa array[4] berisi karakter „\0“ (null character), padahal dalam konstanta string di atas tidak ada karakter tersebut. Dalam bahasa C, null character digunakan untuk menandakan akhir dari sebuah string.

Contoh pendeklarasian string (2):

```
#include <stdio.h>

int main()
{
    // Deklarasi String (2)
    char array[10];

    return 0;
}
```

Contoh di atas akan mendeklarasikan string bernama array yang dapat menampung maksimal 10 karakter, termasuk null character.

Untuk menerima inputan string dari user, kita dapat menggunakan scanf atau gets. Perintah scanf akan membaca inputan string dari user dan berhenti ketika ada spasi, enter (newline) ataupun interupsi dari pengguna. Sedangkan gets akan membaca satu baris kumpulan karakter hingga enter atau interupsi dari pengguna.

Contoh source code penggunaan scanf untuk membaca string:

```
#include <stdio.h>

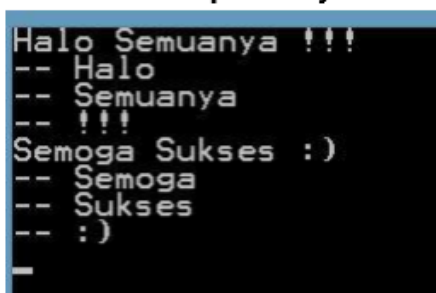
int main()
{
    // Deklarasi array of char dengan kapasitas 10
    char arr[10];

    // Masukkan dan Keluarkan arr selama tidak ada interupsi
    while(true)
    {
        // Menginputkan string kedalam arr
        // PERHATIKAN, bahwa input string berbeda dengan input lainnya
        // tidak perlu menambahkan ampersand (&)
        scanf("%s", arr);

        // Tampilkan string yang telah diinput
        printf("-- %s\n", arr);
    }

    return 0;
}
```

Contoh di atas apabila dijalankan:



```
Halo Semuanya !!!
-- Halo
-- Semuanya
-- !!!
Semoga Sukses :)
-- Semoga
-- Sukses
-- :)
_
```

Contoh penggunaan gets untuk membaca string:

```
#include <stdio.h>

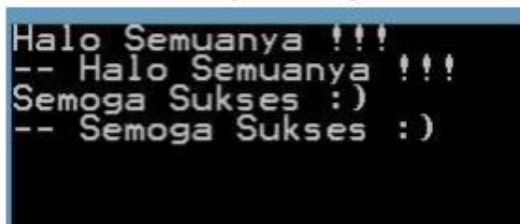
int main()
{
    // Deklarasi array of char dengan kapasitas 100
    char arr[100];

    // Masukkan dan Keluarkan arr selama tidak ada interupsi
    while(true)
    {
        // Menginputkan string kedalam arr dengan menggunakan gets
        gets(arr);

        // Tampilkan string yang telah diinput
        printf("-- %s\n", arr);
    }

    return 0;
}
```

Contoh di atas apabila dijalankan:



```
Halo Semuanya !!!
-- Halo Semuanya !!!
Semoga Sukses :)
-- Semoga Sukses :)
```

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

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, yaitu:

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

Berikut adalah fungsi-fungsi yang dibagi berdasarkan kegunaannya dalam mengolah sebuah string (diambil dari www.cplusplus.com):

1. Copying :
 - a. **memcpy** (Copy block of memory)
 - b. **memmove** (Move block of memory)
 - c. **strcpy** (Copy string)
 - d. **strncpy** (Copy characters from string)
2. Concatenation :
 - a. **strcat** (Concatenate strings)
 - b. **strncat** (Append character from string)

3. Comparison :

- a. **memcmp** (Compare two blocks of memory)
- b. **strcmp** (Compare two strings)
- c. **strcoll** (Compare two strings using locale)
- d. **strncmp** (Compare characters of two strings)
- e. **strxfrm** (Transform string using locale)

4. Searching :

- a. **memchr** (Locate character in block of memory)
- b. **strchr** (Locate first occurrence of character in string)
- c. **strcspn** (Get span until character in string)
- d. **strpbrk** (Locate characters in string)
- e. **strrchr** (Locate last occurrence of character in string)
- f. **strspn** (Get span of character set in string)
- g. **strstr** (Locate substring)
- h. **strtok** (Split string into tokens)

5. Other :

- a. **memset** (Fill block of memory)
- b. **strerror** (Get pointer to error message string)
- c. **strlen** (Get string length)

a. Fungsi strcpy

```
char * strcpy ( char * destination, const char * source );
```

Fungsi strcpy digunakan untuk melakukan copy dari sebuah string ke string lainnya. Contoh penggunaan dalam kode program:

```
#include <stdio.h>
// Library string.h digunakan untuk fungsi strcpy
#include <string.h>

int main()
{
    // Deklarasi kan string a dan b
    char a[] = "Halo";
    char b[10];

    // Copy string a ke string b
    strcpy(b, a);

    // Outputkan string b
    printf("%s\n", b);

    return 0;
}
```


b. Fungsi strcat

```
char * strcat ( char * destination, const char * source );
```

Fungsi strcat digunakan untuk melakukan penempelan sebuah string pada akhir suatu string tertentu. Contoh penggunaan dalam kode program:

```
#include <stdio.h>
// Library string.h digunakan untuk fungsi strcat
#include <string.h>

int main()
{
    // Deklarasi kan string a, b, dan c
    char a[] = "Halo";
    char b[] = " Kawan";
    char c[20];

    // Copy string a ke string c
    strcpy(c, a);

    // Tempelkan string b ke akhir string c
    strcat(c, b);

    // Outputkan string c
    printf("%s\n", c);

    return 0;
}
```

c. Fungsi strcmp

```
int strcmp ( const char * str1, const char * str2 );
```

Fungsi strcmp digunakan untuk melakukan perbandingan sebuah string dengan string yang lain. Return value dari fungsi ini dapat berupa bilangan negatif, nol ataupun positif. Jika fungsi ini mengembalikan nilai negatif, maka *str1* memiliki tingkat leksikografi lebih kecil dari *str2*. Sedangkan jika fungsi ini mengembalikan nilai positif, maka *str1* memiliki tingkat leksikografi lebih besar dari *str2*. Terakhir, jika return value nya nol, maka *str1* sama dengan *str2*. Berikut adalah contoh penggunaan fungsi ini dalam kode program:

```
#include <stdio.h>
// Library string.h digunakan untuk fungsi strcmp
#include <string.h>

int main()
{
    // Deklarasi kan string a, b dan c
    char a[] = "Halo";
    char b[] = "Hai";
    char c[] = "Halo";

    // Cek apakah string a sama dengan b
    if(strcmp(a, b) == 0) printf("String a sama dengan b\n");
    else printf("String a tidak sama dengan b\n");

    // Cek apakah string a sama dengan c
    if(strcmp(a, c) == 0) printf("String a sama dengan c\n");
    else printf("String a tidak sama dengan c\n");

    return 0;
}
```

d. Fungsi strlen

```
size_t strlen ( const char * str );
```

Fungsi strlen digunakan untuk mengetahui panjang dari sebuah string.

```
#include <stdio.h>
// Library string.h digunakan untuk fungsi strlen
#include <string.h>

int main()
{
    // Deklarasi kan string a
    char a[] = "Halo";

    // Outputkan panjang string a
    printf("Panjang string a adalah %d\n", strlen(a));

    return 0;
}
```