TUGAS PENDAHULUAN
MODUL I
PERKENALAN MIKU, *COMPILER* BAHASA C, DAN STANDARD INPUT-OUTPUT

Prepared by: Bagus Hanindhito (20-02-2015)

Problem 1 : Hello World!

Definisi Masalah

Pada *problem* ke-1 ini, kita akan mencoba melakukan penulisan ke *standard input – output*. STDIN atau *standard input stream* merupakan sumber data *default* dari sebuah aplikasi. Sebagian besar aplikasi memiliki sumber *standard input stream* yang berasal dari keyboard walaupun pada modul selanjutnya sumber *input stream* dapat berupa file eksternal. STDOUT atau *standard output stream* merupakan tujuan output dari aplikasi. Sebagian besar aplikasi menggunakan *text console* di layar sebagai *output* dari aplikasi. STDERR atau *standard error stream* merupakan tujuan output dari aplikasi untuk mengeluarkan pesan galat. Sama dengan STDOUT, STDERR biasanya menggunakan *text console* di layar.

Dalam bahasa C terdapat *library* standar yang menyediakan interaksi dengan STDIN dan STDOUT. *Library* ini bernama `stdio.h` dan harus di-*include* dalam program bahasa C kita. Beberapa fungsi yang tersedia dalam *library* ini adalah sebagai berikut.

Function	Purpose
Formatted Input/Output Functions	
`fprintf`	Formatted File Write
`fscanf`	Formatted File Read
`printf`	Formatted Write
`scanf`	Formatted Read
`sprintf`	Formatted String Write
`sscanf`	Formatted String Read
`vfprintf`	Formatted File Write Using Variable Argument List
`vprintf`	Formatted Write Using Variable Argument List
`vsprintf`	Formatted String Write Using Variable Argument List
`File Operation Functions`	
`fclose`	Close File
`fflush`	Flush File Buffer
`fopen`	Open File
`freopen`	Reopen File
`remove`	Remove File
`rename`	Rename File
`setbuf`	Set Buffer (obsolete)
`setvbuf`	Set Buffer
`tmpfile`	Create Temporary File
`tmpnam`	Generate Temporary File Name
`Character Input/Output Functions`	
`fgetc`	Read Character from File
`fgets`	Read String from File

fputc	Write Character to File
fputs	Write String to File
getc	Read Characters from File
getchar	Read Character
gets	Read String
putc	Write Character to File
putchar	Write Character
puts	Write String
ungetc	Unread Character
Block Input/Output Functions	
fread	Read Block from File
fwrite	Write Block to File
File Positioning Functions	
fgetpos	Get File Position
fseek	File Seek
fsetpos	Set File Position
ftell	Determine File Position
rewind	Rewind File
Error Handling Functions	
clearerr	Clear Stream Error
feof	Test for End-of-File
ferror	Test for File Error
perror	Print Error Message

Pada bagian ini, kita akan mencoba membuat aplikasi sangat sederhana yaitu *hello world*. Aplikasi ini cukup mencetak ke STDOUT sebuah kalimat "Hello World!". Berikut ini adalah potongan kode aplikasi tersebut. Lengkapi kode tersebut dengan identitas (*header file*) yang sesuai (lihat petunjuk teknis praktikum).

```
#include <stdio.h>

int main (void)
{
	// Deklarasi Variabel
	printf("Hello World!\n");
	// Algoritma
	return 0;
}
```

Kompilasi kode tersebut dengan GCC lalu jalankan dan lihat hasilnya. Untuk menjalankan *executable file*, gunakan *command prompt* pada Windows lalu berpindah ke direktori tempat *executable file* berada. Kemudian, tulis nama *executable file* tersebut lalu tekan Enter.

Contoh Input dan Output

Input ke STDIN

```
```

Output ke STDOUT

```
Hello World!
```

Deliverable

Simpan tugas *problem* ke-1 dengan nama `problem1.c`. Jangan lupa memberikan identitas (*header file*) di awal file ini. Pastikan program dapat dikompilasi dan dijalankan dengan benar.

Problem 2 : Hello World! dengan Greetings Nama Panggilan

Definisi Masalah

Pada *problem* ke-2 ini, kita akan memodifikasi program pada *problem* ke-1 sehingga program akan menampilkan "Hello World!" diikuti dengan sebuah kata yang diberikan oleh pengguna, misalkan nama panggilan seseorang. Berikut ini adalah potongan kode aplikasi tersebut. Lengkapi kode tersebut dengan identitas (*header file*) yang sesuai (lihat petunjuk teknis praktikum).

```
#include <stdio.h>

int main (void)
{
	// Deklarasi Variabel
	char nama_orang[32];
	// Algoritma
	printf("Masukkan Nama Anda:\n");
	scanf("%s", nama_orang);
	printf("Hello World, %s!\n",nama_orang);
	return 0;
}
```

Kompilasi kode tersebut dengan GCC lalu jalankan dan lihat hasilnya. Untuk menjalankan *executable file*, gunakan *command prompt* pada Windows lalu berpindah ke direktori tempat *executable file* berada. Kemudian, tulis nama *executable file* tersebut lalu tekan Enter.

Contoh Input dan Output

Input ke STDIN

```
Miku
```

Output ke STDOUT

```
Masukkan Nama Anda:
Hello World, Miku!
```

Deliverable

Simpan tugas *problem* ke-2 dengan nama `problem2.c`. Jangan lupa memberikan identitas (*header file*) di awal file ini. Pastikan program dapat dikompilasi dan dijalankan dengan benar.

Problem 3 : Hello World! dengan Greetings Nama Lengkap

Definisi Masalah

Pada *problem* ke-3 ini, kita akan memodifikasi program pada *problem* ke-2 sehingga program akan

menampilkan "Hello World!" diikuti dengan sebuah kalimat yang diberikan oleh pengguna, misalkan nama lengkap seseorang. Kode program tidak diberikan. Anda cukup memodifikasi dari soal sebelumnya dari sisi *formatter* `scanf` atau Anda dapat menggunakan fungsi dari `stdio.h` yang lain.

Kompilasi kode tersebut dengan GCC lalu jalankan dan lihat hasilnya. Untuk menjalankan *executable file*, gunakan *command prompt* pada Windows lalu berpindah ke direktori tempat *executable file* berada. Kemudian, tulis nama *executable file* tersebut lalu tekan Enter.

Contoh Input dan Output

Input ke STDIN

```
Hatsune Miku
```

Output ke STDOUT

```
Masukkan Nama Anda:
Hello World, Hatsune Miku!
```

Deliverable

Simpan tugas *problem* ke-3 dengan nama `problem3.c`. Jangan lupa memberikan identitas (*header file*) di awal file ini. Pastikan program dapat dikompilasi dan dijalankan dengan benar.

Problem 4 : Penjumlahan Dua Bilangan Bulat

Definisi Masalah

Pada *problem* ke-4 ini, program menerima dua buah bilangan bulat (*integer*) dari pengguna. Program kemudian menjumlahkan kedua bilangan ini dan menampilkan hasilnya di *console*. Berikut ini adalah potongan kode aplikasi tersebut. Lengkapi kode tersebut dengan identitas (*header file*) yang sesuai (lihat petunjuk teknis praktikum).

```c
#include <stdio.h>

int main (void)
{
	// Deklarasi Variabel
	int Angka_1;
	int Angka_2;
	int Result;
	// Algoritma
	printf("Masukkan bilangan ke-1:\n");
	scanf("%d",&Angka_1);
	printf("Masukkan bilangan ke-2:\n");
	scanf("%d",&Angka_2);
	Result = Angka_1 + Angka_2;
	printf("Hasil penjumlahan: %d\n",Result);
	return 0;
}
```

Kompilasi kode tersebut dengan GCC lalu jalankan dan lihat hasilnya. Untuk menjalankan *executable file*, gunakan *command prompt* pada Windows lalu berpindah ke direktori tempat *executable file* berada.

Kemudian, tulis nama *executable file* tersebut lalu tekan Enter. Kesalahan yang paling sering muncul adalah pada saat memanggil scanf, variabel yang diberikan berupa *address* (*passing by reference*) bukan nilai (*passing by value*). Untuk mendapatkan *address* dari sebuah variabel, digunakan tanda `&` di depan variabel tersebut. Hal ini akan dibahas lebih lanjut pada modul praktikum mengenai pointer.

Contoh Input dan Output

Input ke STDIN

```
10
20
```

Output ke STDOUT

```
Masukkan bilangan ke-1:
Masukkan bilangan ke-2:
Hasil penjumlahan: 30
```

Deliverable

Simpan tugas *problem* ke-4 dengan nama `problem4.c`. Jangan lupa memberikan identitas (*header file*) di awal file ini. Pastikan program dapat dikompilasi dan dijalankan dengan benar.

Problem 5 : Penjumlahan Dua Bilangan Real

Definisi Masalah

Pada *problem* ke-5 ini, program menerima dua buah bilangan real (*float*) dari pengguna. Program kemudian menjumlahkan kedua bilangan ini dan menampilkan hasilnya di *console*. Modifikasi program pada *problem* ke-4. Ubah tipe variabel yang digunakan untuk menampung kedua bilangan yang akan dijumlahkan dan hasilnya. Ubah juga *formatter* `scanf` agar dapat memformat bilangan real. Tampilkan hasil penjumlahan dengan dua angka di belakang koma

Kompilasi kode tersebut dengan GCC lalu jalankan dan lihat hasilnya. Untuk menjalankan *executable file*, gunakan *command prompt* pada Windows lalu berpindah ke direktori tempat *executable file* berada. Kemudian, tulis nama *executable file* tersebut lalu tekan Enter.

Contoh Input dan Output

Input ke STDIN

```
9.9999999999
22.2222
```

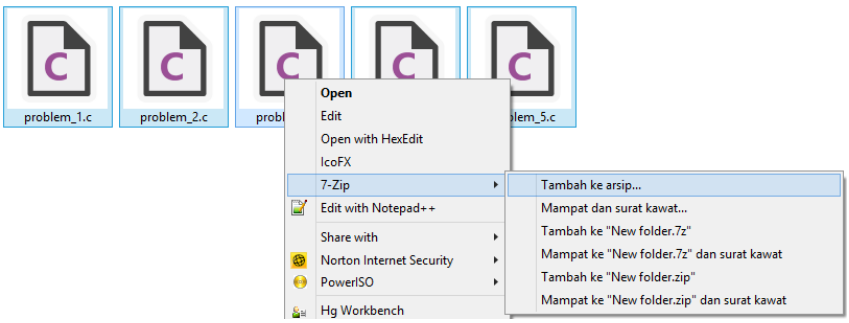
Output ke STDOUT

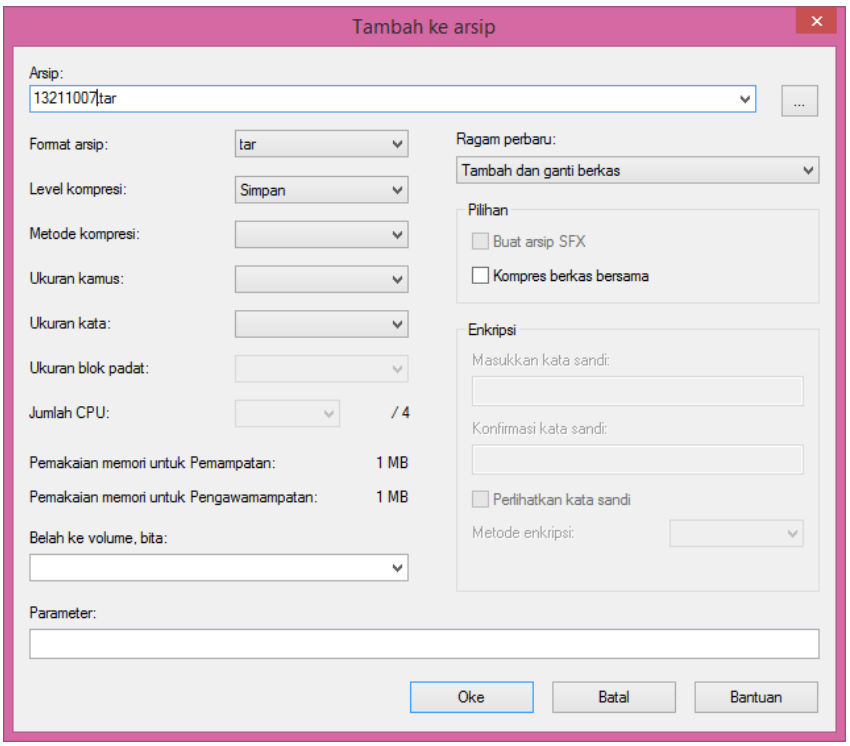
```
Masukkan bilangan ke-1:
Masukkan bilangan ke-2:
Hasil penjumlahan: 32.22
```

Simpan tugas *problem* ke-5 dengan nama `problem5.c`. Jangan lupa memberikan identitas (*header file*) di awal file ini. Pastikan program dapat dikompilasi dan dijalankan dengan benar.

Petunjuk Penyerahan Tugas Pendahuluan Modul I

Simpan kelima file (`problem1.c`, `problem2.c`, `problem3.c`, `problem4.c`, dan `problem5.c`) dalam satu folder. Gunakan program 7-zip untuk mengkompresi menjadi arsip TAR (.tar). Penamaan file TAR bebas (disarankan menggunakan NIM). File TAR ini yang akan di-submit ke server MIKU saat pengumpulan tugas pendahuluan saat memasuki laboratorium. Hanya file kode saja yang dimasukkan ke dalam arsip TAR. File *executable* tidak perlu dimasukkan.





Selesai