PROSEDUR 1

Asisten Dasar Pemrograman Genap 2021/2022

PROSEDUR 1

Tujuan:

1. Praktikan dapat mengetahui apa itu prosedur
2. Praktikan dapat mengerti dengan baik konsep dari prosedur
3. Praktikan dapat mengimplementasikan prosedur sesuai dengan kebutuhan

Landasan Teori

Prosedur merupakan sebuah modul program yang digunakan untuk mengerjakan suatu hal yang spesifik. Prosedur digunakan untuk membantu *programmer* agar dapat mengerjakan suatu hal berulang maupun tidak berulang dengan lebih mudah. Terdapat juga beberapa permasalahan yang hanya dapat diselesaikan dengan penggunaan konsep-konsep dari prosedur.

Keuntungan dalam menggunakan prosedur adalah:

1. Code menjadi lebih terstruktur
2. Proses Debugging menjadi lebih mudah
3. Efisiensi Code (Tidak perlu mengetik code yang sama berulang kali untuk masalah yang sama)
4. Jika bekerja dalam Tim, Prosedur dapat dengan mudah diatur dan digunakan oleh rekan tim.

Karakteristik dari Prosedur:

1. Prosedur menggunakan keyword (*Syntax*) **void**
2. Prosedur biasanya dinamai dengan sebuah **kata kerja** {tampilData, editData, hapusData,…}
3. Prosedur dapat memiliki sebuah parameter maupun tidak
4. Prosedur memiliki "**Lingkungan**"-nya sendiri, sehingga terdapat konsep *local* dan *global variabel.*
5. Umumnya, prosedur yang baik tidak memiliki *standard input/output*

PENGGUNAAN PROSEDUR

Silahkan amati code yang menggunakan Prosedur dan tidak menggunakan Prosedur.

```
20
21      string user1;
22      int umurUser1;
23
24      string user2;
25      int umurUser2;
26
27      strcpy(user1, "Budi");
28      umurUser1 = 30;
29
30      strcpy(user2, "Anto");
31      umurUser2 = 28;
32
33      printf("Daftar User : ");
34      printf("\n\tUsername : %s", user1);
35      printf("\n\tUmur     : %d", umurUser1);
36      printf("\n");
37      printf("\n\tUsername : %s", user2);
38      printf("\n\tUmur     : %d", umurUser2);
39
40      return 0;
41  }
42
```

Pada contoh disamping, kita hanya memiliki 2 user. Andaikan saja teman-teman memiliki 5 user atau lebih. Maka teman-teman akan terus menerus menulis **baris code yang sama.**

Teman-teman akan menulis code baris 27-31 dan baris 34-38 **berulang kali**.

Hal ini menyebabkan pemborosan dan tentu saja code akan sangat susah untuk di ***maintance***

```
7
8   void inputData(string user, int *umur, string tempUser, int tempUmur){
9       strcpy(username, tempUser);
10      *umur = tempUmur;
11  }
12
13  void tampilData(string username, int umur){
14      printf("\n\tUsername : %s", username);
15      printf("\n\tUmur     : %d", umur);
16      printf("\n");
17  }
18
19  int main() {
20
21      string user1;
22      int umurUser1;
23
24      string user2;
25      int umurUser2;
26
27      inputData(user1, &umurUser1, "Budi", 30);
28      inputData(user2, &umurUser2, "Anto", 28);
29
30      printf("Daftar User : ");
31      tampilData(user1, umurUser1);
32      tampilData(user2, umurUser2);
33
34      return 0;
35  }
36
```

Mendefinisikan Prosedur

Pemanggilan Prosedur

Sedangkan jika teman-teman menggunakan Prosedur, maka teman-teman hanya perlu **memanggil prosedur** yang telah teman-teman buat.

Sehingga meskipun teman-teman memiliki 5 user atau lebih, teman-teman tidak perlu repot-repot untuk menuliskan baris code yang sama.

DEKLARASI, DEFINISI, DAN PEMANGGILAN:

Prosedur memiliki 3 bagian penting, yakni : Nama, Parameter, dan Body. untuk template dari Prosedur adalah sebagai berikut:

```
void <namaProsedur> ( <Parameter> ) {
      <Body>
}
```

Penjelasan :

1. namaProsedur → digunakan untuk mengindentifikasi dan memanggil prosedur
2. Parameter → digunakan untuk memberikan inputan agar dapat digunakan di dalam prosedur
3. Body → merupakan isi dari prosedur bersangkutan.

Nama dari prosedur harus ***unik.*** Hal ini dikarenakan nama dari prosedur akan **mengidentifikasi** prosedur mana yang ingin kita gunakan. Sehingga jika teman-teman memiliki 2 prosedur atau lebih, sistem dapat mengetahui dengan jelas prosedur mana yang ingin teman-teman gunakan.

Parameter digunakan untuk memberikan inputan tambahan yang diperlukan. Sebuah prosedur **dapat memiliki parameter ataupun tidak**. Oleh karena itu, parameter bersifat *optional* pada sebuah prosedur.

Body adalah ***hal apa yang dilakukan ketika prosedur dipanggil***. Body tentu saja menjadi sebuah keharusan. Karena jika tidak, maka prosedur akan dianggap tidak melakukan apa-apa dan hanya akan memberatkan memori.

```
 8
 9  void sapaUser(string nama){
10
11      printf("Hello %s\n", nama);
12
13  }
14
15  void berikanPesan(){
16      printf("Selamat Belajar Bro/Sist\n");
17  }
18
19  int main() {
20
21      string namaUser;
22      printf("Input Nama : "); fflush(stdin); gets(namaUser);
23
24      sapaUser(namaUser);
25      berikanPesan();
26
27      return 0;
28  }
```

Pemanggilan Prosedur

Pada contoh, terlihat 2 prosedur, yakni:

sapaUser dengan parameter berupa **<string> nama**

berikanPesan yang **tidak memiliki Parameter**.

Memanggil parameter dilakukan dengan mengetikkan nama Prosedur disertai dengan tanda kurung '()'

Ketika memanggil *sapaUser*, maka kita perlu **menginput parameter yang dibutuhkan**. Sedangkan jika memanggil *berikanPesan* maka kita **tidak perlu memberikan inputan parameter**

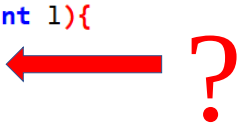
Teman-teman juga dapat memanggil prosedur di dalam prosedur lainnya. Asalkan teman-teman harus mendefinisikan ataupun mendeklarasikan fungsi yang ingin dipanggil di atas terlebih dahulu. Silahkan perhatikan contoh berikut.

```c
void printLuas(int p, int l){
    printf("\nLuas     : %d", p*l);
}

void printKeliling(int p, int l){
    printf("\nKeliling : %d", 2*(p+l));
}

void printDetail(int p, int l){
    printLuas(p,l);
    printKeliling(p,l);
}

int main() {
    int panjang = 10;
    int lebar   = 20;

    printDetail(panjang, lebar);
    return 0;
}
```

Bisa dilihat pada contoh, ketika teman-teman menjalankan program disamping. Maka program akan **berjalan seperti biasa**.

printDetail akan memanggil prosedur *printLluas* dan *printKeliling*. Dikarenakan *printLuas* dan *printKeliling* telah **didefinisi** di atas *printDetail*. Maka *printDetail* tidak akan mengalami kesusahan dalam memanggil prosedur yang dibutuhkan.

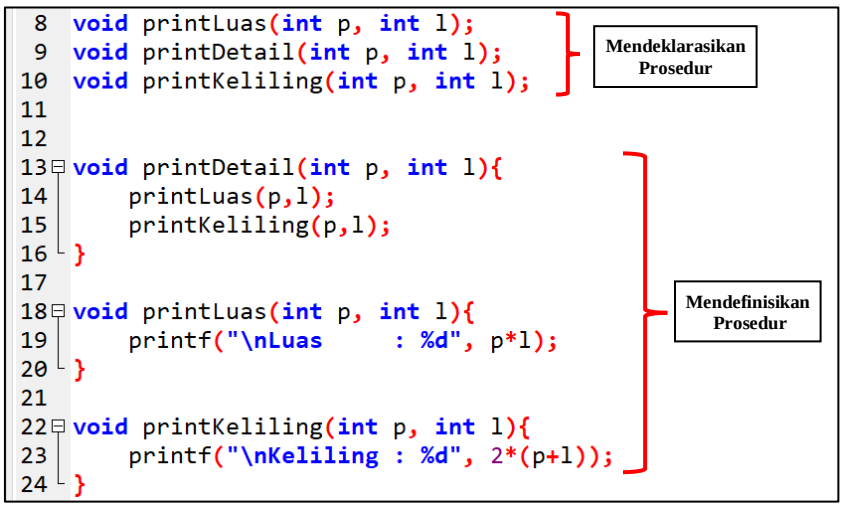
```c
void printDetail(int p, int l){
    printLuas(p,l);
    printKeliling(p,l);
}

void printLuas(int p, int l){
    printf("\nLuas     : %d", p*l);
}

void printKeliling(int p, int l){
    printf("\nKeliling : %d", 2*(p+l));
}

int main() {
    int panjang = 10;
    int lebar   = 20;

    printDetail(panjang, lebar);
    return 0;
}
```



Pada contoh ini, program akan mengeluarkan Warning. Hal ini dikarenakan *printDetail* tidak menemukan **Body** dari *printLuas* dan *printKeliling*. Sehingga program akan mengeluarkan **Warning**.

Hal ini dapat dicegah dengan cara **mendeklarasikan** terlebih dahulu prosedur-prosedur yang akan kita gunakan.

```
Message
[Warning] conflicting types for 'printLuas'
[Note] previous implicit declaration of 'printLuas' was here
[Warning] conflicting types for 'printKeliling'
[Note] previous implicit declaration of 'printKeliling' was here
```



Dengan mendeklarasikan prosedur terlebih dahulu, maka semua **warning** tersebut akan hilang.

Teman-teman diharapkan untuk mendeklarasikan terlebih dahulu prosedur yang digunakan agar tidak terjadi ***Undefined Behavior*** pada program teman-teman.

Dengan mendeklarasikan prosedur, teman-teman juga dapat dengan mudah melihat apa saja prosedur yang telah teman-teman buat.

PARAMETER PROSEDUR:

Dikarenakan Paramater prosedur bersifat optional, maka prosedur dapat untuk tidak memiliki parameter. Sebagai contohnya adalah sebuah prosedur untuk menampilkan menu dari program yang kita buat. Untuk memanggil prosedurnya, maka teman-teman **diharuskan agar tidak menginputkan parameter apapun.**

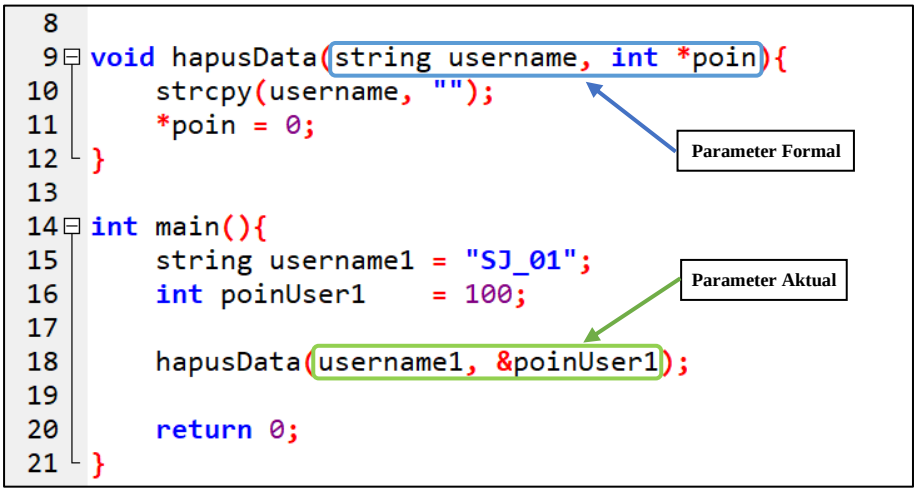
```
 8  void tampilMenu(){
 9      printf("\n\t---- Menu ----");
10      printf("\n[1] Registrasi Menu");
11      printf("\n[2] Edit Data");
12      printf("\n[3] Tampil Data");
13  }
14
15  int main() {
16      tampilMenu();
17  }
```

Namun terkadang kita membutuhkan sebuah tambahan informasi dari luar Prosedur. Sebagai contoh, prosedur untuk menghitung **jumlah pendapatan** membutuhkan informasi berupa inputan **harga barang**, **jumlah barang terjual**, dan **harga beli barang** bersangkutan. cara mendapatkan informasi-informasi tersebut adalah dengan menggunakan **parameter**. Sehingga informasi-informasi dapat dimasukkan ke dalam prosedur (*passing)* dan dapat digunakan oleh Prosedur.

```
 8  void hitungPendapatan(int jumlah, float harga, float beli){
 9      float untung = jumlah * (harga-beli);
10
11      printf("Pendapatan Bersih : %.2f", untung);
12  }
13
14  int main() {
15      int jumlahBarang;
16      float hargaJual, hargaBeli;
17
18      printf("Input Jumlah Jual: "); scanf("%d", &jumlahBarang);
19      printf("Input Harga  Jual: "); scanf("%f", &hargaJual);
20      printf("Input Harga  Beli: "); scanf("%f", &hargaBeli);
21
22      hitungPendapatan(jumlahBarang, hargaJual, hargaBeli);
23      return 0;
24  }
```

Perlu diingat bahwa kita mendeklarasikan **3 parameter** di prosedur *hitungPendapatan* pada contoh. Sehingga, saat ingin memanggil prosedur tersebut, maka teman-teman harus menginputkan 3 parameter juga sesuai dengan tipe data-nya masing-masing. Jika tidak, maka teman-teman akan mendapatkan ***error*** maupun ***warning***. Kesimpulannya, jumlah parameter **Formal** (Parameter pada pendeklarasian Prosedur) dan jumlah parameter **Aktual** (Parameter yang kita inputkan saat memanggil Prosedur) harus sama.

Parameter dapat dibedakan menjadi 2, yakni **Parameter Formal** dan **Parameter Aktual**. Parameter Formal merupakan sebutan bagi Parameter yang kita **definisikan pada Prosedur**. Sedangkan Parameter Aktual merupakan sebutan bagi **variabel yang kita *passing*** ke dalam Prosedur.




```
 8
 9  void hapusData(string username, int *poin){
10      strcpy(username, "");
11      *poin = 0;
12  }
13
14  int main(){
15      string username1 = "SJ_01";
16      int poinUser1    = 100;
17
18      hapusData(username1, &poinUser1);
19
20      return 0;
21  }
```

Berikut adalah karakteristik dari Parameter Formal dan Parameter Aktual:

1. Parameter Formal dan Parameter Aktual **tidak** harus memiliki **nama** yang sama
2. Parameter Formal dan Parameter Aktual harus memiliki **tipe data** yang sama
3. **Jumlah** Parameter Aktual dan Parameter Formal harus sama.
4. **Urutan** Parameter Formal harus sama dengan Parameter Aktual

```
 8
 9
10  void cariSelisih(string nama, float nilai){
11
12      printf("Nama  User : %s"  , nama );
13      printf("Nilai User : %.2f", nilai);
14
15  }
16
17  int main(){
18
19      string namaUser = "Vince";
20      float nilaiUser = 3.9;
21
22      cariSelisih(namaUser, nilaiUser)
23      void cariSelisih (string nama, float nilai)
24      return 0;
25  }
26
```

Pada Contoh, dapat dilihat nama dari **variabel Formal** maupun **Variabel Aktual** dapat berbeda, seperti yang terlihat pada kotak berwarna **Hijau.**

Tipe data dari **Parameter Aktual** juga harus sama dengan **Parameter Formal.** Urutan saat diberikan juga harus disesuaikan. Jika tidak, maka program akan memberikan **warning** atau membuat **Program Error**

NAÏVE, SEMI-NAÏVE OUTPUT, SEMI-NAÏVE INPUT, NETT EFFECT

Prosedur dapat dibedakan menjadi 4 jenis, yakni **Naïve, Semi-Naïve Input, Semi-Naïve Output,** dan **Nett Effect.** Perbedaan diantara keempat jenis prosedur tersebut terdapat pada parameter yang digunakan.

- Naïve

  Merupakan prosedur yang tidak memiliki Parameter sama sekali.

```
 9  void tampilMenu(){
10      printf("\n\t---- Menu ----");
11      printf("\n[1] Menu 1");
12      printf("\n[2] Menu 2");
13      printf("\n[3] Menu 3");
14      printf("\n[4] Menu 4");
15  }
16
```

- Semi-Naïve Input

  Merupakan Prosedur yang hanya memiliki parameter Input dan menghasilkan output yang dikeluarkan melalui Standard I/O

```
 8
 9  void hitungKeliling(int pjg, int lbr){
10
11      int keliling = 2 * (pjg+lbr);
12      printf("Keliling : %d", keliling);
13
14  }
15
16
```

- Semi-Naïve Output

  Merupakan Prosedur yang hanya memiliki parameter Output dan menggunakan input yang didapat melalui Standard I/O

```
 8
 9  void hitungKeliling(int *keliling){
10
11      int pjg, lbr;
12      printf("Input Panjang : "); scanf("%d", pjg);
13      printf("Input Lebar   : "); scanf("%d", lbr);
14
15      *keliling = 2 * (pjg+lbr);
16  }
17
```

- Nett Effect

  Merupakan Prosedur yang memiliki Parameter *Input-Output.* Dan tidak menggunakan Standard I/O. lebih lengkap dijelaskan di Modul **Prosedur 2**

Sekilas Mengenai Pointer

Jika teman-teman perhatikan, pada contoh kita terkadang menggunakan **pointer** dan terkadang juga menggunakan sebuah parameter biasa. Ketika teman-teman menggunakan **pointer,** maka ketika terjadi perubahan di Parameter Formal, maka **Parameter Aktual akan ikut berubah**. Sebaliknya, ketika teman-teman menggunakan **parameter biasa**, maka perubahan pada Parameter Formal **tidak akan mengakibatkan perubahan di Parameter Aktual**.

Untuk lebih memahami perbedaan penggunaan Pointer dan Parameter Biasa, diharapkan teman-teman dapat mengcopy-paste code di bawah ini dan memperhatikan perubahan nilai yang terjadi.

```
 8  void prosedurPointer(int *temp){
 9      *temp = 100;
10  }
11
12  void prosedurBiasa(int temp){
13      temp = 50;
14  }
15
16  int main() {
17      int tempA = 1;
18      int tempB = 2;
19
20      printf("\nTemp A Sebelum : %d"  , tempA);    // 1
21      printf("\nTemp B Sebelum : %d\n", tempB);    // 2
22      prosedurPointer(&tempA);
23      prosedurBiasa(tempB);
24      printf("\nTemp A Sesudah : %d", tempA);      // 100
25      printf("\nTemp B Sesudah : %d", tempB);      // 2
26
27      return 0;
28  }
```

Parameter Pointer dan **Parameter Biasa** akan dibahas lebih lanjut di Modul Selanjutnya.

Guided

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <time.h>

typedef char string[50];

void tampilMenu();
void setBilangan(int *bilanganAsli, int inputan);
void tampilOperasi(string operasi, int angka1, int angka2, int hasil);

void prosedurPenjumlahan(int *hasil, int input1, int input2);
void prosedurPengurangan(int *hasil, int input1, int input2);
void laporanOperasi(bool sudahJumlah, bool sudahKurang, int hasilJumlah, int hasilKurang);

void inputBilanganN(int *temp);
void tebakAngka(int *inputan);

```

```c
int main() {

    srand(time(NULL));

    int menu;
    int temp;
    int i;

    bool sudahInputBilangan = false;
    bool sudahPenjumlahan = false;
    bool sudahPengurangan = false;

    int bilangan1;
    int bilangan2;
    int hasilPenjumlahan;
    int hasilPengurangan;

    bool sudahInputN = false;
    int inputan;
    int n;
```

```c
    do{
        tampilMenu();
        printf("\n>>> "); scanf("%d", &menu);

        switch(menu){

            case 1 :
                printf("\n\t\t--- Input Bilangan ---");

                do{
                    printf("\n\tInput Bilangan Pertama : "); scanf("%d", &temp);
                    if(temp < 0){
                        printf("\t\t\t[!] Inputan Tidak Boleh < 0\n");
                    }else{
                        break;
                    }
                }while(true);

                setBilangan(&bilangan1, temp);


```

```
63                  do{
64                      printf("\n\tInput Bilangan Kedua   : "); scanf("%d", &temp);
65                      /*
66                          perhatikan if ini sama persis dengan if diatas
67                          kita bisa saja menjadikan ini prosedur
68                          tetapi akan lebih tepat menggunakan metode lain (fungsi)
69                          sehingga untuk sementara lebih baik diginikan dulu ya :)
70                      */
71                      if(temp < 0){
72                          printf("\t\t\t[!] Inputan Tidak Boleh < 0\n");
73                      }else{
74                          break;
75                      }
76
77                  }while(true);
78
79                  setBilangan(&bilangan2, temp);
80                  sudahInputBilangan = true;
81                  printf("\n\t\t[*] Berhasil Menginput Bilangan");
82              break;
83
```

```
84          case 2 :
85              if(!sudahInputBilangan){
86                  printf("\n\t\t[!] Belum Input Bilangan");
87              }else{
88
89                  printf("\n\t\t---- Penjumlahan ----");
90                  prosedurPenjumlahan(&hasilPenjumlahan, bilangan1, bilangan2);
91
92                  //mungkin sedikit "tricky" disini
93                  //tetapi ini salah satu variasi penggunaan prosedur
94                  tampilOperasi("Penjumlahan", bilangan1, bilangan2, hasilPenjumlahan);
95
96                  //perlu diketahui bahwa tidak akan menjadi masalah besar
97                  //apabila teman-teman menulis manual seperti ini
98
99                  /*
100                     printf("\n\tPenjumlahan %d dan %d", bilangan1, bilangan2);
101                     printf("\n\tHasilnya adalah       : %d", hasilPenjumlahan);
102                 */
103                 sudahPenjumlahan = true;
104             }
105         break;
```

```
107         case 3 :
108             if(!sudahInputBilangan){
109
110                 printf("\n\t\t[!] Belum Input Bilangan");
111
112             }else{
113
114                 printf("\n\t\t---- Pengurangan ----");
115                 prosedurPengurangan(&hasilPengurangan, bilangan1, bilangan2);
116
117                 tampilOperasi("Pengurangan", bilangan1, bilangan2, hasilPengurangan);
118
119                 /*
120                     printf("\n\tPengurangan %d dan %d", bilangan1, bilangan2);
121                     printf("\n\tHasilnya adalah       : %d", hasilPengurangan);
122                 */
123                 sudahPengurangan = true;
124             }
125         break;
126
127
```

```
130         case 4 :
131
132             if(!sudahInputBilangan){
133                 printf("\n\t\t[!] Belum Input Bilangan");
134             }else{
135
136                 printf("\n\t\t--- Laporan ---\n");
137
138                 laporanOperasi(sudahPenjumlahan, sudahPengurangan, hasilPenjumlahan, hasilPengurangan);
139
140             }
141
142         break;
143
144         case 5 :
145             inputBilanganN(&n);
146
147             sudahInputN = true;
148
149         break;
150
```

```
152         case 6 :
153
154             if(!sudahInputN){
155                 printf("\n\t\t[!] Belum Menginputkan N");
156             }else{
157
158                 for(i=0 ; i<n ; i++){
159
160                     tebakAngka(&inputan);
161                     printf("\tKamu Menginputkan : %d\n", inputan);
162
163                 }
164
165             }
166         break;
167
168         case 0 :
169             printf("\n\t[*] Program Keluar");
170             printf("\n\t    <Nama Praktikan> - <NPM>");
171         break;
172
```

```
172
173
174             default:
175
176                 printf("\n\t[!] Menu Tidak Tersedia");
177
178             break;
179
180         }
181
182         getch();
183
184     }while(menu != 0);
185
186     return 0;
187
188 }
189
190
191
192
```

```
194 void tampilMenu(){
195 
196     system("cls");
197 
198     printf("\n\t--- Guided ---");
199     printf("\n[1] Input Bilangan");
200     printf("\n[2] Penjumlahan   ");
201     printf("\n[3] Pengurangan   ");
202     printf("\n[4] Laporan       ");
203     printf("\n-------------------");
204     printf("\n[5] Input N               ");
205     printf("\n[6] Input Sebanyak N Kali ");
206     printf("\n[0] Keluar Program        ");
207 }
208 
```

```
209 /*
210     prosedur yang baik sejatinya simple
211     dan melaksanakan tugas yang spesifik
212     seperti prosedur berikut
213 */
214 void setBilangan(int *bilanganAsli, int inputan){
215     *bilanganAsli = inputan;
216 }
217 
218 void prosedurPenjumlahan(int *hasil, int input1, int input2){
219     *hasil = input1 + input2;
220 }
221 
222 void prosedurPengurangan(int *hasil, int input1, int input2){
223     *hasil = input1 - input2;
224 }
225 
226 void tampilOperasi(string operasi, int angka1, int angka2, int hasil){
227     printf("\n\t%s %d dan %d", operasi, angka1, angka2);
228     printf("\n\tMenghasilkan : %d", hasil);
229 }
```

```
231 /*
232     prosedur dibawah ini merupakan prosedur yang kompleks
233     pada kasus umum tidak disarankan menuliskan prosedur yang terlalu kompleks
234     kecuali jika memang diharuskan atau telah di-break ke prosedur-prosedur
235     yang lebih kecil
236 */
237 void laporanOperasi(bool sudahJumlah, bool sudahKurang, int hasilJumlah, int hasilKurang){
238 
239     printf("\n\tHasil Penjumlahan Terakhir : ");
240     if(sudahJumlah){
241         printf("%d", hasilJumlah);
242     }else{
243         printf("<Belum Ada>");
244     }
245 
246     printf("\n\tHasil Pengurangan Terakhir : ");
247     if(sudahKurang){
248         printf("%d", hasilKurang);
249     }else{
250         printf("<Belum Ada>");
251     }
252 
253 }
```

```
255
256 void inputBilanganN(int *temp){
257
258     //note bahwa ini adalah variabel lokal
259     //sehingga tidak bertabrakan dengan
260     //variabel menu di main();
261     int menu;
262
263     system("cls");
264     printf("\n\t--- Input N ---");
265     printf("\n[1] Dari User    ");
266     printf("\n[2] Random [1-10]");
267     printf("\n>>> "); scanf("%d", &menu);
268
269     switch(menu){
270         case 1 :
271             do{
272                 //perhatikan disini kita tidak menggunakan $ pada scanf();
273                 //jika parameter berupa pointer (ada tanda *)
274                 //maka scanf tidak menggunakan tanda reference ($)
275
276                 printf("\n\tInput N : "); scanf("%d", temp);
277
278                 //alternatif lainnya adalah
279                 //scanf("%d", &(*temp));
280
281                 if(*temp < 0 ){
282                     printf("\t\t[!] Tidak Boleh Negatif\n");
283                 }else{
284                     break;
285                 }
286             }while(true);
287
288             printf("\n\t\t[*] Berhasil Input N");
289         break;
290
291         case 2 :
292
293             //ingat rumus (rand() % (maxRange-MinRange+1)) + minRange
294             *temp = (rand() % (10-1+1) + 1);
295             printf("\n\t\t[*] Berhasil Input N");
296             printf("\n\t\t    Nilai N : %d", *temp);
297         break;
298
299         default:
300             printf("\n\t\t[!] Menu Tidak Tersedia");
301             printf("\n\t\t    Kembali ke Menu Awal");
302         break;
303     }
304 }
305
306 void tebakAngka(int *inputan){
307
308     int temp;
309     printf("\n\tTebak Bilangan : "); scanf("%d", &temp);
310
311     /*
312         apa yang bisa kamu lakukan disini? :)
313     */
314
315     *inputan = temp;
316 }
317
318
319
```

Ketentuan & Format Pengumpulan Guided:

1. Untuk comment tidak menjadi masalah jika tidak ditulis di Guided
2. Teman-teman boleh berkreasi dengan Guided seperti mengganti nama variabel, membuat prosedur lain, mengganti logika program, dll. Dengan ketentuan
    - Program bisa di-compile tentunya
    - Fungsionalitas Program tetap terjaga
    - Memiliki **minimal 8 prosedur**
3. **Khusus di modul saya** (**Prosedur 1**). Saya mengizinkan untuk menuliskan definisi prosedur di file terpisah. Tetapi tetap harus dikumpulkan. Ketentuan ini juga berlaku di UGD dan Tugas nanti. Untuk modul selanjutnya silahkan ditanyakan ke pemegang modul bersangkutan.
4. Tidak diperkenankan untuk menggunakan modul-modul selanjutnya yaaa… [Fungsi, Array, Record]

5. Semua kode dimasukkan ke dalam sebuah folder dengan format penamaan: **GD7_X_YYYY**
6. Folder tadi kemudian di Zip dengan format penamaan: **GD7_X_YYYYY.zip**
7. Keterangan:
    - X = Kelas
    - YYYY = 5 digit terakhir NPM Praktikkan
    - Z = Nomor Soal

HINT UGD & Tugas(Bonus)

- Perlu diketahui Guided menu 1 – 4 ditujukan agar teman-teman lebih mengenal **prosedur** dan mulai terbiasa dengan **prosedur**. Tantangan terbesarnya bagi teman-teman adalah mengetahui kapan menggunakan pointer (*) dan kapan tidak menggunakan pointer. Simplenya jika variabel tersebut akan berubah di dalam prosedur dan perubahan tersebut **penting** di main(), maka gunakan Pointer. Jika tidak, maka tidak perlu menggunakan pointer.
- Menu 1 – 4 **tidak** memiliki keterkaitan erat dengan UGD, akan tetapi **Konsep** dari menu 1 – 4 akan sangat ber-keterkaitan dengan UGD dan akan sangat membantu. Sooo…. pastikan teman-teman mengerti dengan benar konsepnya (baik di main(); maupun prosedur-nya).
- Menu 5 dan 6 merupakan menu yang paling mirip dengan UGD.
- Teman-teman boleh berkreasi nih dengan menu 6. Apa yang teman-teman dapat lakukan? Menambah nilai random? Memeriksa apakah nilai inputan sebelum lebih besar dari nilai inputan sekarang? Kalau mencari rata-rata dari inputan bagaimana? Apakah inputan user mendekati suatu bilangan? Dan sebagainya.. :)