Laporan Kegiatan Pengabdian Pada Masyarakat

Pembinaan dan Pelatihan Olimpiade Sains Kabupaten/Kota Bidang Pemrograman Tingkat SMA Di SMAN 3 Semarang

Oleh

Wijanarto., M.Kom.

Fakultas Ilmu Komputer
Universitas Dian Nuswantoro
Semarang 2013

Pengabdian pada masyarakat ini atas permintaan dan di biayai oleh pihak SMA Negeri 3 Semarang, berdasarkan surat permohonan nomer : 423.7/173/2013 serta berdasarkan surat tugas Dekan Nomer :128/B.18.01/UDN-02/IV/2013

KATA PENGANTAR

Syukur ke hadirat Alloh SWT kami haturkan atas selesainya laporan pengabdian pelatihan OSK di SMA 3 Semarang yang sudah berlangsung pada tanggal 18 – 21 Pebruari 2013. Salah satu bentuk dari tri darma perguruan tinggi adalah mengabdikan tenaga dan pikiran seorang dosen ke masyarakat, selain melakukan pengajaran di lingkungan PT dan melakukan penelitian.

Kegiatan ini dapat berlangsung dengan lancar atas dukungan dari segenap civitas akademika SMA Negeri 3 Semarang khususnya guru pembina tim olimpiade dan laboran pada laboratorium komputer, serta dukungan kepala sekolah pada umumnya.

Pengabdian yang dilakukan penulis saat ini adalah memberikan pelatihan di bidang pemrograman dan algoritma pada siswa SMA 3 Semarang, yang di bentuk oleh lingkungan setempat guna di persiapkan untuk terjun mengikuti kopetisi Olimpiade Informatika pada tingkat Kabupaten dan Kota pada tahun 2013. Namun di balik tujuan jangka pendek tersebut, tujuan utama dari pengabdian ini adalah memberi bekal kepada siswa untuk dapat berpikir anaitik dan lebih menyukai bidang pemrograman yang sudah menjadi trebd setter saat ini.

Tak ada gading yang tak retak, laporan ini di susun dengan harapan penulis dapat melakukan perbaikan kepada diri sendiri yang hasilnya akan di terapkan pada masyarakat luas. Segala kritik dan saran dengan tangan terbuka penuis akan terima, akhirnya dengan segala kerendahan hati, penulis memohon maaf jika ketidak sempurnaan dalam laporan ini menggangu pembaca yang budiman.

Semarang, Pebruari 2013

Wijanarto

DAFTAR ISI

HALAMAN COVER i
KATA PENGANTAR ii
DAFTAR ISI iii
I. PENDAHULUAN.
A. ANALISA SITUASI 1
B. PERUMUSAN MASALAH 2
II. TUJUAN DAN MANFAAT
A. TUJUAN 2
B. MANFAAT 3
III. PELAKSANAAN KEGIATAN
A. MATERI PELATIHAN 3
B. WAKTU PELAKSANAAN 3
C. JADUAL PELAKSANAAN 4
D. PESERTA 4
E. ALAT BANTU 5
F. METODE PENGAJARAN 5
IV. HASIL KEGIATAN 5
V. KESIMPULAN DAN SARAN
A. KESIMPULAN 6
B. SARAN 6

LAMPIRAN –LAMPIRAN

LAMPIRAN 1. SURAT PERMOHONAN PELATIHAN
LAMPIRAN 2. SURAT TUGAS DEKAN
LAMPIRAN 3. DAFTAR HADIR PELATIHAN
LAMPIRAN 4. DOKUMENTASI PELATIHAN
LAMPIRAN 5. MATERI
LAMPIRAN 6. LATIHAN MENULIS PROGRAM
LAMPIRAN 7. CONTOH SOAL DAN PEMBAHASAN

I. Pendahuluan

A. Analisa Situasi

Seiring kemajuan teknologi secara umum, dan khususnya di bidang teknologi informatika (Komputer), mempersiapkan siswa untuk menguasai hal tersebut merupakan tantangan yang harus di jawab oleh para pendidik kita. Mulai dari tingkat dasar hingga perguruan tinggi, computer sudah menjadi hal yang lumrah untuk di kuasai. Terlebih dewasa ini sudahdi adakan kontes di bidang computer,baik di tingkat siswa menengah, hingga perguruan tinggi, baik dalam skala local, regional, nasional bahkan internasional.

Untuk menghadapi persaingan dan kemajuan teknologi computer, khususnya di bidang pemrogrman, terlebih lagi untuk mempersiapkan siswa untuk dapat terjun ke dalam suatu kontes, maka perlu diadakan suatu pelatihan, baik secara dasar, mengengah dan lanjut, dengan tujuan siswa akan dapat mengikuti perkembangan kemajuan sekaligus berprestasi di bidang informatika melalui kontes atau olimpiade yang sudah sering dan menjadi agenda tahunan di Negara kita ini.

Olimpiade Sain Kabupaten/Kota, khususnya di bidang Informatika,, merupakan bentuk kegiatan rutin yang di selenggarakan oleh tiap kabupaten/Kota hingga propinsi secara nasional, untuk dapat di ikutkan dalam kompetisi olimpiade internasional. SMA Negeri 3 Semarang dalam rangka menangkap peluang tersebut mempersiapkan siswa siswinya dalam mengkuti kegiatan tersebut. Namun karena mata ajar pemrograman tidak ada dalam kurikulum sekolah menengah atas di indonesia, maka pengajaran hanya di berkikan dalam bentuk ekstra kurikuler saja. Dengan demikian kedalaman materi untuk mengikuti kontes olimpiade menjadi sangat kurang. Hal ini yang menyebabkan kegiatan ini muncul, karena pihak sekolah menginginkan pembinaan, pelatihan yang di sajikan oleh pengajar yang lebih fokus dan lebih memahami persoaan pemrograman. Dengan mengajuan permohonan ke Fakultas Ilmu Komputer Universitas Dian Nuswantoro, Semarang, maka pihak sekolah memang sudah tepat melakukan langkah pembinaan internal dengan mendatangkan pembina eksternal. Gayung bersambut, karena kepentingan

tersebut memberi efek pada kedua belah pihak, pembinaan yang di lakukan sebagai upaya pengabdian yang di lakukan oleh dosen dalam menyebarkan ilmu sehingga bermanfaat secara langsung, di pihak lain sekolah memerlukan pembinaan intensif guna menghadapi OSK.

Akhirnya, semoga pelatihan ini dapat memberi bekal bagi siswa kita untuk lebih termotivasi dan giat mempelajari bidang pemrograman ini dengan lebih baik dan benar, karena pada dasarnya kegiatan ini memang hanya pemicu awal dari perjalanan panjang siswa untuk mempelajari dunia ilmu komputer secara umum dan ilmu pemrograman secara khusus.

B. Perumusan Masalah

Berdasarkan analisa situasi di atas maka, dapat di rumuskan permasalahan sebagai berikut, Bagaimana meningkatkan kemampuan ketrampilan dan analitik siswa dalam mengerjakan soal-soal Olimpiade Sain Kabupaten/Kota (OSK) untuk mengikuti kompetisi OSK di Jawa Tengah.

II. Tujuan Dan Manfaat

A. Tujuan

Secara umum, pelatihan ini bertujuan mempersiapkan siswa dalam menghadapi olimpiade sain informatika di tingkat lokal (kabupaten dan provinsi) , akan tetapi secara khusus dalam bidang pemrograman computer lebih di tujukan untuk :

a. Melatih siswa menulis program secara benar, dari aspek tata cara menulis kode yang “terbaca” sehingga mudah di telusuri.
b. Mengenalkan tipe data (dari tipe data dasar hingga array), dan bagaimana melakukan manipulasi terhadap tipe tersebut
c. Melakukan analisis dasar permasalahan yang di wakili oleh tipe data yng sudah di pelajari
d. Mempelajari teknik melakukan input-proses-output pemrograman dalam skala kecil dan sedang, mengolah data bilangan, karakter dan string.
e. Menyelesaikan suatu masalah yang di wakili oleh permasalahan di bidang matematis dasar (pada tingkat kontes biasanya, masalah yang muncul adalah bidang matematika).

f. Mengenalkan algoritma-algoritma sederhana untuk penyelesaian suatu masalah (statistic sederhana, pertukaran data, traversal data)

B. Manfaat

Secara jelah maka manfaat dari pembinaan dan pelatihan Olimpiade ini memberi efek langung pada ketrampilan penulisan dan pembuatan program, serta kemampuan analitik dalam menjawab persoalan yang sifatnya algoritmis.

III. Pelaksanaan kegiatan

A. Materi Pelatihan (Terlampir)

Materi pelatihan ini disesuaikan dengan tingkat pemahaman dasar dan tujuan pelatihan itu sendiri, karena pelatihan ini di tujukan untuk mempersiapkan peserta mengikuti olimpiade sain informatika, maka materi di sesuaikan dengan tujuan kontes tersebut. Secara umum materi di bagi atas peningkatan pemahaman di bidang :

.1. Tipe Data dasar
.2. Input dan output dasar
.3. Struktur dasar pemrograman
.4. Mengolah Larik (satu dan dua dimensi)
.5. Parsing data secara efektif dari alat inputan
.6. Prosedur dan Fungsi
.7. Rekursif

No	Materi	Durasi
1	Tipe Data Dasar	2 Jam (Teori/Praktek)
2	Input Dan Output Dasar	4 Jam (Teori/Praktek)
3	Struktur dasar Pemrograman	6 Jam (Teori/Praktek)
4	Larik	6 Jam (Teori/Praktek)
5	Parsing	2 jam (Teori/Praktek)
6	Prosedur dan Fungsi	6 Jam (Teori/Praktek)
7	Rekursif	6 Jam (Teori/Praktek)
	Total	32 Jam

B. Waktu pelaksanaan

Waktu Pelaksanaan kegiatan pengabdian pada masyarakat secara umum di laksanakan pada waktu kegiatan sekolah jeda dan kegiatan dosen jeda :

Tanggal : 18 – 21 Pebruari 2013

Tempat : Laboratorium SMA Negeri 3 Semarang

C. Jadual pelaksanaan

Jadual pelaksanaan pembinaan dan pelatihan sebagai berikut :

Tanggal	Jam	Tempat	Materi
18-02-2013	08.00 - 09.00	Lab. SMAN 3	Pembukaan
	09.30 - 11.40	Lab. SMAN 3	Tipe Data (T/P/E)
	12.00 - 13.00	Lab. SMAN 3	ISHOMA
	13.15 - 14.30	Lab. SMAN 3	I/O(T/E)
	14.30 - 15.00	Lab. SMAN 3	ISHOMA
	15.00 - 16.00	Lab. SMAN 3	I/O (P)
19-02-2013	08.00 - 09.00	Lab. SMAN 3	SDA (T/E)
	09.30 - 11.40	Lab. SMAN 3	SDA (P)
	12.00 - 13.00	Lab. SMAN 3	ISHOMA
	13.15 - 14.30	Lab. SMAN 3	Latihan SDA
	14.30 - 15.00	Lab. SMAN 3	ISHOMA
	15.00 - 16.00	Lab. SMAN 3	Larik (T/E)
20-02-2013	08.00 - 09.00	Lab. SMAN 3	Larik (P)
	09.30 - 11.40	Lab. SMAN 3	Parsing Larik (T/P/E)
	12.00 - 13.00	Lab. SMAN 3	ISHOMA
	13.15 - 14.30	Lab. SMAN 3	Proc. Func. (T/E)
	14.30 - 15.00	Lab. SMAN 3	ISHOMA
	15.00 - 16.00	Lab. SMAN 3	Proc. Func (P)
21-02-2013	08.00 - 09.00	Lab. SMAN 3	Lat. Proc.Func.
	09.30 - 11.40	Lab. SMAN 3	Recc (T)
	12.00 - 13.00	Lab. SMAN 3	ISHOMA
	13.15 - 14.30	Lab. SMAN 3	Recc.(T/E)
	14.30 - 15.00	Lab. SMAN 3	ISHOMA
	15.00 - 16.00	Lab. SMAN 3	Evaluasi (E)

Keterangan : Semua Kegiatan di lakukan oleh satu pembina/Pelatih : Wijanarto.,M.Kom. T: Teori, P: Praktek, E: Evaluasi bahas soal latihan

D. Perserta

Peserta pelatihan di tujukan untuk siswa menengah, dari berbagai kelas yang mempunyai ketertarikan dan minat di bidang pemrograman, untuk di persiapkan mengikuti olimpiade sain di tingkat lokal (kabupaten dan provinsi). Demi efektifitas pelatihan, maka jumlah peserta untuk setiap kelas akan di batasi maksimal 20 siswa dengan satu instruktur, jika lebih maka akan di buatkan kelas tersendiri.Peserta pelatihan dan pembinaan adalah siswa yang di usulkan oleh pihak SMA Negeri 3 Semarang (terlampir di absensi).

E. Alat Bantu (tools)

Alat bantu utama dalam pelatihan ini adalah satu unit komputer (desktop atau laptop) yang sudah terinstall program aplikasi *Free Pascal*. Di pilihnya jenis bahasa pascal, di karenakan bahasa ini sudah terstruktur dengan baik, sehingga lebih mudah bagi siswa untuk mempelajari pemrograman prosedural. Sedangkan produk yang di pilih adalah *Free Pascal*, karena alasan produk tersebut adalah bebas pakai tanpa adanya ikatan ataupun lisensi komersial.

F. Metode Pengajaran

Semua materi di atas akan di sajikan dalam bentuk presentasi, baik dengan slide ataupun di ajarkan secara langsung di dalam kelas. Sistem pelatihan dan pengajaran adalah pemberian pemahaman teori dan dilanjutkan dengan praktikum secara langsung oleh instruktur. Metode ini sangat cocok untuk pelatihan, karena pada pelatihan ini prosentasi materi yang di berikan adalah 75% praktikum dan 25% pembahasan teroritis. Pada materi teoritis instruktur akan menerangkan ide dasar dari setiap materi, bagaimana memanfaatkannya, dengan contoh sederhana dan memberi problem untuk di pecahkan para peserta. Instruktur dapat mendemonstrasikan suatu permasalahan dan penyelesaiannya dengan program, TANPA memberikan source code ke peserta. Instruktur hanya menuliskan notasi algoritma yang mendekati bahasa yang di pakai (kecuali saat menerangkan reserved words yang di pakai dalam bahasa tersebut) untuk menerangkan satu problem. Pada sesi praktikum siswa harus selalu di beri problem untuk di selesaikan sesuai materi yang di berikan pada sesi tersebut. Semua problem disesuaikan dengan kemampuan peserta, dan merupakan representasi dari bidang matematika (statistik dasar, keprimaan, deret, vektor, matrik, kombinatorik) dalam berbagai tipe data.

IV. Hasil Kegiatan

Hasil kegiatan pelatihan ini berupa ketrampilan peserta dalam melakukan parsing data input dan output, serta lebih memahami dan memaknai soal kompetisi olimpiade. Selain itu perserta dapat lebih cepat mengena`li pola soal pemrograman

khususnya mengenali struktur data yang di pakai setelah mengikuti pelatihan. Tidak ada wujud yang tampak dari pelatihan ini tetapi terukur melalui kecepatan siswa dalam mengerjakan soal latihan, walaupun pretest dan post-test tidak di lakukan uji beda secara stastitik.

V. Kesimpulan dan Saran

A. Kesimpulan

Secara umum baik peserta pelatihan dan pembina merasa cukup puas dengan meningkatan skill siswa serta bertambahnya pengetahuan yang ti dak di dapatkan sebelumnya mengenai teknik mengerjakan soal olimpiade. Bertambahnya pengetahuan struktur data membantu siswa lebih cepat beradaptasi dan mengerjakan soal dengan tepat. Teknik atau algoritma yang di ajarkan membantu siswa lebih cepat dalam mencari solusi untuk persoalan yang rumit.

B. Saran

Kedepan perlu di jadwalkan program serupa yang berkesinambungan dan bertahap, tidak saja di berikan kepada siswa yang akan mengikuti lomba olimpiade, tetapi di tujukan untuk seluruh siswa yang mengikuti ekstrakurikuler pemrogaman sebagai media pencarian bibit baru di bidang pemrograman untuk didik secara lebih baik.

LAMPIRAN-LAMPIRAN

LAMPIRAN 1. SURAT PERMOHONAN PELATIHAN

PEMERINTAH KOTA SEMARANG
DINAS PENDIDIKAN
SMA NEGERI 3 SEMARANG
Jalan Pemuda 149 Telp. (024) 3544287 – 3544291 Semarang
Website : www.sman3-smg.sch.id Email : kepala_sma3smg@yahoo.co.id

Nomor : 423.7/173/2013
Lamp : -
Hal : **Permohonan Pembinaan Dosen**
Fakultas Ilmu Komputer (FIK) UDINUS

Kepada Yth.
Kepala Program Study
Fakultas Ilmu Komputer (FIK) UDINUS
Di Semarang

Kami beritahukan dengan hormat bahwa, dalam rangka pembinaan kelas olympiade SMA Negeri 3 Semarang yang akan menghadapi lomba.

Berkenaan dengan itu agar siswa lebih menguasai materi lomba dengan baik kami mohon kepada kepala program study Fakultas Ilmu Komputer (FIK) UDINUS berkenan mengijinkan :

Nama : **Wijanarto, M.Kom**
NPP : 0686.11.2009.354
Jabatan : Dosen Fakultas Ilmu Komputer (FIK) UDINUS

Untuk memberikan pembinaan dan bimbingan selama 4 hari pada :

Hari : Senin - Kamis
Tanggal : 18 – 21 Februari 2013
Pukul : 08.00 s.d. selesai
Tempat : **Lab. Komputer SMA N 3 Semarang**

Besar harapan kami atas berkenan kepala program study Fakultas Ilmu Komputer (FIK) UDINUS mengijinkan Bp. Wijanarto, M.Kom memberikan pembinaan dan bimbingan kepada siswa kelas olympiade SMA Negeri 3 Semarang.

Demikian atas perhatiannya kami sampaikan banyak terima kasih.

Semarang, 16 Februari 2013
Kepala Sekolah,

Drs. H. Bambang Nianto Mulyo, M.Ed.
NIP 196104291986031007

LAMPIRAN 2. SURAT TUGAS DEKAN

FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO

SURAT TUGAS

No : 128/B.18.01/UDN-02/IV/2013

Dekan Fakultas Ilmu Komputer Universitas Dian Nuswantoro memberikan tugas kepada :

Nama : Wijanarto, M.Kom

Untuk memberikan pembinaan dan bimbingan kelas olympiade SMA Negeri 3 Semarang yang akan menghadapi lomba, pada:

Hari	: Senin - Kamis
Tanggal	: 18 – 21 Februari 2013
Pukul	: 08.00 - Selesai
Tempat	: Lab. Komputer SMA N 3 Semarang
Lain-lain	: 1. Harap dilaksanakan dengan sebaik-baiknya dan penuh tanggung jawab. 2. Memberikan laporan setelah melaksanakan tugas.

Semarang, 15 Februari 2013

Dr. Abdul Syukur
NIP. 0686.11.1992.017

Yang bersangkutan telah melaksanakan tugas dengan baik pada tanggal .....
.....2013, di .....

Tanda tangan dan stempel

- (Sebagai bukti lampirkan deskripsi kegiatan yang telah dilaksanakan)

LAMPIRAN 3. DAFTAR HADIR PELATIHAN

DAFTAR HADIR SISWA

PEMBINAAN OLYMPIADDE KOMPUTER

SMA NEGERI 3 SEMARANG

NAMA	TANDA TANGA
	1
[illegible] Agfin	2
[illegible]	3
[illegible]	4
Agu Nabila	5
[illegible] Muhlas A [illegible]	6
[illegible]	7
[illegible]	8
[illegible]	9
[illegible]	10
[illegible]	11
[illegible]	12
[illegible]	13
[illegible]	14
[illegible]	15

Semarang, 18 Februari 2013

Mengetahui,
Koordinator Olympiade

[illegible], M.Si
[illegible]031000

Pembina Olympiade Kompu.er

R SMA BI

M. Khanif, M. Kom
NIP.197802072006041018

DAFTAR HADIR SISWA

PEMBINAAN OLYMPIADDE KOMPUTER

SMA NEGERI 3 SEMARANG

NO	NAMA	TANDA TANGA
1	[illegible]	1
2	Ardhan Arifin	2
3	[illegible]	3
4	[illegible]	4
5	Putri Ayu Nabila	5
6	[illegible]	6
7	[illegible]	7
8	[illegible]	8
9	[illegible]	9
10	[illegible]	10
11	[illegible]	11
12		12
13		13
14		14
15		15

Semarang, 15 Februari 2013

Mengetahui,
Koordinator Olympiade

Drs. Subiyanto, M.Si
NIP.196006181987031009

Pembina Olympiade Komputer

R SMA BI

M. Khanif, M. Kom
NIP.197802072006041018

DAFTAR HADIR SISWA

PEMBINAAN OLYMPIADDE KOMPUTER

SMA NEGERI 3 SEMARANG

NO	NAMA	TANDA TANGA
1	[illegible]	1
2	Muhammad Mubtar A bnor	2
3	[illegible]	3
4	[illegible]	4
5	Putri Ayu Nabila	5
6	[illegible]	6
7	[illegible]	7
8	[illegible]	8
9	[illegible]	9
10	[illegible]	10
11	[illegible]	11
12		12
13		13
14		14
15		15

Semarang, 20 Februari 2013

Mengetahui,
Koordinator Olympiade

Drs. Sabiyanto, M.Si
NIP.19680618198703100[illegible]

Pembina Olympiade Komputer

M. Khanif, M. Kom
NIP.197802072006041018

DAFTAR HADIR SISWA

PEMBINAAN OLYMPIADDE KOMPUTER

SMA NEGERI 3 SEMARANG

NO	NAMA	TANDA TANGA
1	Angga Andno P	1
2	Author Ayton	2
3	Muhammad Faiz Zidane	3
4	[illegible]	4
5	Putri Ayu Nabila	5
6	Muhammad Muhlas Abror	6
7	[illegible]	7
8	[illegible]	8
9	[illegible]	9
10	[illegible]	10
11	[illegible]	11
12	[illegible]	12
13	[illegible]	13
14	[illegible]	14
15		15

Semarang, 21 Februari 2013

Mengetahui,
Koordinator Olympiade

Drs. Subiyanto, M.Si
NIP.196006181987031000

Pembina Olympiade Komputer

M. Khanif, M. Kom
NIP.197802072006041018

LAMPIRAN 4. DOKUMENTASI PELATIHAN

Dokumentasi Pelatiahan OSK
Di SMA 3 Semarang 18 – 21 Pebruari 2013

Pemakalah dan Pembina OSK/OSN SMA 3 Semarang

Peserta Kelompok Kelas 10 (18-19 Pebruari 2013)

Peserta Kelompok Kelas 11 (20-21 Pebruari 2013)

LAMPIRAN 5. MATERI

Algoritma dan Pemrograman

WIJANARTO

PELATIHAN DAN PEMBEKALAN PEMROGRAMAN UNTUK OLIMPIADE SAIN KOTA/KABUPATEN/PROPINSI

Materi

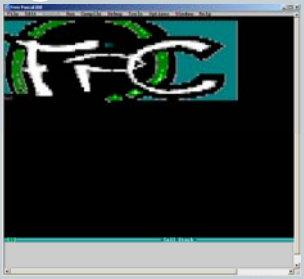
- Free Pascal
- Tipe Data dasar
- Input dan output dasar
- Struktur dasar pemrograman
- Mengolah Larik (satu dan dua dimensi)
- Parsing data secara efektif dari alat inputan
- Prosedur dan Fungsi
- Rekursif

Referensi

- Lecturer Note In Computer Science, Pascal User Manual Report, Kathleen Jensen and Niclaus Wirth
- The Art Of Computer Programming, Vol 1. Fundamental Algorithms, D.E. Knuth
- Program=Data Structure+Algorithm, N. Wirth
- The Programming Language Pascal, N. Wirth

Free Pascal

- Open Source
- Prosedural
- Terstruktur



- http://www.freepascal.org/download.var

Kerangka Dasar Program

```
Program Nama(input output);
{uses
  type
  const
  var
   fungsi/ prosedur}
Begin
  {algoritma}
End.
```

Indentasi sangat penting
Supaya program "Terbaca"

Vocabulary

B. Table of word-delimiters (reserved words)

and	end	nil	set
array	file	not	then
begin	for	of	to
case	function	or	type
const	goto	packed	until
div	if	procedure	var
do	in	program	while
downto	label	record	with
else	mod	repeat	

Vocabulary

+	:	(
-	,	)
*	=	[
/	<>	]
:=	<	{
.	<=	}
,	>=	↑
;	>	..

Vocabulary

A. Table of standard identifiers

Constants:
false, true, maxint

Types:
integer, Boolean, real, char, text

Program parameters:
input, output

Functions:
abs, arctan, chr, cos, eof, eoln, exp, ln, odd,
ord, pred, round, sin, sqr, sqrt, succ, trunc

Procedures:
get, new, pack, page, put, read, readln, reset,
rewrite, unpack, write, writeln

Konsep Data

- Numerik

Numbers			
Data Type	Bytes	Number type	Range
byte	1	integer	0..255
shortint	1	integer	-127..128
word	2	integer	0..65535
smallint	2	integer	-32767..32768
longword	4	integer	0..4294967295
longint	4	integer	-2147483648 .. 2147483647
qword	8	integer	0 .. 18446744073709551615
int64	8	integer	-9223372036854775808 .. 9223372036854775807
integer	2, 4 or 8	integer	same as smallint, longint or int64
cardinal	2, 4 or 8	integer	same as word, longword or qword
real	4 or 8	floating point	system dependent
single	4	floating point	1.5E-45 .. 3.4E38, 7-8 significant digits
double	8	floating point	5.0E-324 .. 1.7E308, 15-16 significant digits
extended	10	floating point	1.9E-4951 .. 1.1E4932, 19-20 significant digits
comp	8	floating point	-2E64+1 .. 2E63-1, 19-20 significant digits
currency	8	fixed point	-922337203685477.5808..922337203685477.5807 (fixed at 4 digits < 1)

Konsep Data

- Text Dan Karakter

Text and Characters		
Data Type	Size in bytes	Description
char	1	Stores a single ASCII character. i.e. 'a'
string[size]	size	Also known as a shortstring. This is the original Pascal string. It works as an array of characters, with the length of the actual string stored in varname[0].
string	variable	The current default string type is a dynamically allocated array of characters. To get or change the length, use the Length and SetLength procedures. See Lesson 8 for more information on dynamic arrays.
widestring	variable	Same as string, but in Unicode format - that means that each character now takes twice as much space. It also allows you to create multi-lingual applications, since Unicode is required for many international characters.
pchar	4+size of string	pchar is a pointer to a null terminated string. This is useful when working with routines written in other languages, such as C.

Konsep Data

- Boolean (8 bit)
  - True
  - False

Operator dan Expresi

- Expresi di konstruksi secara komputasional untuk mendapatkan nilai dari variabel dan men-generalisasi nilai baru dengan suatu operasi. Expresi terdiri dari operan, misal variabel dan constan, operator, dan function.

```
function Max(a: vector; n: integer): real;
var x: real; i: integer;
begin x := a[1];
      for i := 2 to n do
      begin {x = max(a[1],...,a[i-1])}
          if x < a[i] then x := a[i]
      end;
      {x = max(a[1],...,a[n])}
      Max := x
end
```

Operator dan Expresi

- Operator NOT
  - Negasi dari suatu operan

operator	operation	type of operand(s)	result type
:=	assignment	any type except file types	---
arithmetic:			
+ (unary)	identity	integer or real	same as operand
- (unary)	sign inversion		
+	addition	integer or real	integer or real
-	subtraction		
*	multiplication		
div	integer division	integer	integer
/	real division	integer or real	real
mod	modulus	integer	integer

Operator dan Expresi

relational:			
=	equality	scalar, string, set, or pointer	Boolean
<>	inequality		
<	less than	scalar or string	
>	greater than		
<=	less or equal -or- set inclusion	scalar or string set	
>=	greater or equal -or- set inclusion	scalar or string set	
in	set membership	first operand is any scalar, the second is its set type	

Operator dan Expresi

logical:			
not	negation		
or	disjunction	Boolean	Boolean
and	conjunction		
set:			
+	union		
-	set difference	any set type T	T
*	intersection		

Operator dan Expresi

- Perkalian

operator	operation	type of operands	type of result
*	multiplication set intersection	real, integer any set type T	real, integer T
/	division	real, integer	real
div	division with truncation	integer	integer
mod	modulus	integer	integer
and	logical "and"	Boolean	Boolean

Contoh

Expression	*Value*
10 DIV 2 * 3	15
10 MOD 3 - 4 DIV 2	-1
5 * 2 / 4 * 2	5.0
5 * 2 / (4 * 2)	1.25
5 + 2 / (4 * 2)	5.25

Lowest Precedence	*Highest Precedence*
+	*
-	/
	DIV
	MOD

Operator dan Expresi

- Penjumlahan

operator	operation	type of operands	type of result
+	addition set union	integer, real any set type T	integer, real T
-	subtraction set difference	integer, real any set type T	integer, real T
or	logical "or"	Boolean	Boolean

Operator dan Expresi

- Relasi

operator	type of operands	result
= <> < > <= >=	any scalar or subrange type	Boolean
in	any scalar or subrange type and its set type respectively	Boolean

Fungsi Standar

Arithmetic functions

abs(x) computes the absolute value of x. The type of the result is the same as that of x, which must be either intege or real.

sqr(x) computes x*x. The type of the result is the same as that of x, which must be either integer or real.

sin(x) for the following, the type of x must be either real or integer. The type of the result is always real.

cos(x)
arctan(x)
exp(x)
ln(x) (natural logarithm)
sqrt(x) (square root)

Fungsi Standar

Predicates (Boolean functions)

odd(x)	the type of x must be integer; the result is true if x is odd, otherwise false.
eoln(f)	returns the value true when, while reading the textfile f, the end of the current line is reached; otherwise, false.
eof(f)	returns the value true when, while reading the file f, the "end-of-file" is reached; otherwise, false.

Fungsi Standar

Transfer functions

trunc(x)	x must be of type real; the result is the greatest integer less than or equal to x for x>=0, and the least integer greater or equal to x for x<0.
round(x)	x must be of type real; the result, of type integer, is the value x rounded. That is, round(x) = trunc(x+0.5), for x ≥ 0 trunc(x-0.5), for x < 0
ord(x)	the ordinal number of the argument x in the set of values defined by the type of x.
chr(x)	x must be of type integer, and the result is the character whose ordinal number is x (if it exists).

Fungsi Standar

Further standard functions

succ(x)	x is of any scalar type (except real), and the result is the successor value of x (if it exists).
pred(x)	x is of any scalar type (except real), and the result is the predecessor value of x (if it exists).

Statement

- Tunggal

```
A:=10;
```

- Blok/banyak

```
Begin
 a:=10;b:=20;
 c:=a+b;
End;
```

Input dan output

Routine	Description
append	Opens a file for writing at its end.
close	Closes a file.
filesize	Returns the current size of a file.
flush	Writes the output buffer for the specified Pascal file into the associated operating system file.
getfile	Returns a pointer to the C standard I/O descriptor associated with the specified Pascal file.
linelimit	Terminates program execution after a specified number of lines has been written into a text file.
message	Writes the specified information to stderr.
open	Associates an external file with a file variable.
read and readln	Read in boolean, integer and floating-point variables, fixed- and variable-length strings, enumerated types, and pointers.
remove	Removes the specified file.
reset and rewrite	Accepts an optional second argument.
seek	Resets the current position of a file for random access I/O.
tell	Returns the current position of a file.
write and writeln	Outputs boolean integer and floating-point variables, fixed- and variable-length strings, enumerated types, and pointers; output expressions in octal or hexadecimal; allows negative field widths.

Struktur dasar pemrograman

- Sequence
  - Urutan statemen,ekspresi
- Branch/Conditional
  - Cascade
  - Nested

```
<if statement> ::= if <expression> then <statement> |
    if <expression> then <statement> else <statement>
```

```
<case statement> ::= case <expression> of
  <case list element> {;<case list element>} end
<case list element> ::= <case label list> : <statement> |
                        <empty>
<case label list> ::= <case label> {,<case label> }
```

Contoh

```
If x<1.5 then z:=x+y else z:=1.5
If z>5.0 then
    begin
        y:= z mod 2;
        if y=0 then dec(x) else inc(z)
    end
Else (z div 2) mod 2;
```

```
case operator of
  plus:   x := x+y;
  minus:  x := x-y;
  times:  x := x*y
end
```

```
case i of
   1: x := sin(x);
   2: x := cos(x);
   3: x := exp(x);
   4: x := ln(x)
end
```

Studi Kasus 1

- Input : satu bilangan bulat sembarang
- Output: Menentukan bilangan tersebut genap atau ganjil, prima atau bukan.
- Misal :
- input : 7
- Output : 7 bilangan ganjil dan prima
- input : 8
- Output : 8 bilangan genap dan bukan prima

source

Studi Kasus 2

- Input : 4 bilangan bulat sembarang
- Output: Menentukan bilangan tersebut terbesar atau terkecil, genap atau ganjil, prima atau bukan,
- Misal :
- input : 7 4 5 10
- Output :
  - 4 bilangan genap terkecil , bukan prima.
  - 10 bilangan genap terbesar, bukan prima
- input : 7 12 30 50
- Output :
  - 7 bilangan ganjil terkecil , prima.
  - 50 bilangan genap terbesar, bukan prima

source

Pengulangan atau Loop

- Elemen:
  - **Kondisi pengulangan: ekspresi lojik**
  - **Badan pengulangan: aksi yang diulang**
- Notasi pengulangan:
  1. Berdasarkan jumlah pengulangan
  2. Berdasarkan kondisi berhenti
  3. Berdasarkan kondisi pengulangan
  4. Berdasarkan dua aksi
  5. Berdasarkan pencacah

Pengulangan Berdasarkan Jumlah Pengulangan

- Aksi akan diulang sebanyak n kali, dan bukan urusan pemrogram untuk mengelola pengulangan tersebut
- Dengan hanya menyebutkan pengulangan tersebut, pengulangan pasti akan berhenti suatu saat

```
repeat n times
    Aksi
```

Pengulangan Berdasarkan Kondisi Berhenti

- *Aksi akan dihentikan jika kondisi-berhenti dipenuhi (berharga true), akan diulang jika kondisi-berhenti belum tercapai*
- Badan pengulangan pada notasi ini (*Aksi*) *minimal akan dilakukan satu kali karena pada waktu eksekusi pengulangan*yang pertama tidak dilakukan test terhadap *kondisi-berhenti*
- *Test terhadap kondisi berhenti dilakukan setelah Aksi* Dilaksanakan
- Pengulangan berpotensi mengalami "kebocoran", jika ada kemungkinan bahwa seharusnya *Aksi tidak pernah boleh dilakukan untuk kasus tertentu*

```
repeat
  aksi
until kondisi-berhenti
```

Pengulangan Berdasarkan Kondisi Pengulangan

- *Aksi akan dilakukan selama kondisi-pengulangan masih dipenuhi (berharga true)*
- Test terhadap *kondisi-pengulangan dilakukan setiap kali sebelum aksi dilaksanakan*
- Pengulangan ini berpotensi untuk menimbulkan aksi "kosong" (tidak pernah melakukan apa-apa) karena pada test yang pertama, *kondisi-pengulangan tidak dipenuhi (berharga false)*
  - Badan pengulangan (*aksi*) *pada notasi ini mungkin tidak akan pernah dilakukan, karena sebelum aksi yang pertama dieksekusi* dilakukan test terhadap *kondisi-berhenti*

while (*kondisi-pengulangan*) **do**
aksi

Pengulangan Berdasarkan 2 Aksi

- Pengulangan ini seolah-olah adalah "gabungan" antara bentuk pengulangan kedua dan ketiga
- Mekanisme yang dilakukan oleh pengulangan ini adalah dengan melakukan secara otomatis Aksi-1 pada eksekusi yang pertama kemudian dilakukan test terhadap kondisi berhenti
- Tergantung kepada kondisi berhenti yang ditest:
  - Aksi-2 akan diaktifkan dan kemudian aksi-1 yang berikutnya diulang, atau
  - Pengulangan dihentikan karena efek neto dari aksi-1 menghasilkan kondisi berhenti
- Pengulangan ini berguna untuk kasus-kasus di mana Aksi-2 merupakan hal yang harus dilakukan tergantung dari hasil *Aksi-1*.

```
iterate
        Aksi-1
stop <kondisi-berhenti>
        Aksi-2
```

Pengulangan Berdasarkan Pencacah

- *nama-pencacah harus suatu type yang terdefinisi suksesor dan predesesornya, setelah pelaksanaan pengulangan selesai, harga* yang tersimpan pada nama-pencacah tidak terdefinisi: jika hendak dipakai, harus didefinisikan kembali
- Aksi akan dilakukan dengan memperhitungkan harga-harga dari nama-pencacah yang di-"jelajahi"
- Dengan memakai pengulangan ini, pemrogram tidak perlu melakukan operasi terhadap suksesor/predesesor karena setiap kali selesai melakukan *Aksi, otomatis mesin akan melakukan operasi untuk mendapatkan suksesor dari harga yang berlaku saat itu untuk* Nama
- *range-harga bisa dari kecil ke besar atau sebaliknya*

nama-pencacah **traversal** [*range-harga*]
aksi

Loop

- For
  - For <start> to <end> do
  - For <start> downto <end> do
- While
  - While<kondisi> do
  - End;
- Repeat
  - Repeat
  - Until<kondisi>

For

- Digunakan untuk batas perulangan yang sudah di ketahui
- Dapat di variasi dengan conditional untuk keluar dari perulangan
  - Break conditional
- Variasi lainnya nested akan di bahas di array 2 dimensi

Algoritma	Pascal

i traversal [Awal..Akhir]
Aksi
	/* Jika Awal <= Akhir */
	for (i := Awal to Akhir)
	do
	begin
	Aksi
	end;
	/* Jika Awal >= Akhir */
	for (i := Akhir downto Awal) do
	begin
	Aksi
	end;

contoh

```
For i:=1 to 10 do
      write(i:2);

For i:=1 to 10 do
 begin
    if (i>5) and (i mod 2 <> 0) then
 break;
    write(i:2);
 end;
```

While...do

- Digunakan jika batas perulangan tidak di ketahui
- Dapat di variasi dengan conditional untuk keluar dari perulangan
- Akan memeriksa kondisi, sebelum melakukan aksi

Algoritma	Pascal

while <kondisi-ULANG> do
Aksi
	while (kondisi-ULANG) do
	begin
	Aksi
	end;

repeat Aksi until <kondisi-STOP>	repeat Aksi until (kondisi-STOP);

Contoh

```
i:=1;x:=100;
While (x>i) do
 begin
  write(i:2);
  dec(x);
  i:= (I mod
 x)+inc(i);
End;
```

```
i:=1;x:=100;t:=true;
While (t) do
 begin
  write(i:2);
  dec(x);
  if i=(x div 2) then t:=
  false;
End;
```

Repeat

- Digunakan jika batas perulangan tidak di ketahui
- Dapat di variasi dengan conditional untuk keluar dari perulangan
- Setidaknya akan melakukan suatu aksi, setelah itu baru memeriksa kondisi

Algoritma	Pascal

iterate
Aksi-A
stop <kondisi-STOP>
Aksi-B
	(* deklarasi stop : boolean *)
	stop := false;
	repeat
	Aksi-A;
	if (kondisi-STOP) then
	stop := true
	else
	Aksi-B;
	until (stop);

contoh

```
i:=0;
Repeat
    i:=i+1;
    write(i);
Until(i<=10)
```

```
i:=6
Repeat
  write(i);
  inc(i);
Until(i>5)
```

I/O Lanjut

- Eof memeriksa input device berada di akhir file atau tidak
  - Eof tanpa parameter, memeriksa secara interaktif saat pembacaan input tanpa file.
  - Eof ➔ boolean
  - Eof(file), file adalah file stream buffer
- Eoln memeriksa input device berada di akhir baris atau tidak
  - Eoln tanpa parameter, memeriksa secara interaktif saat pembacaan input tanpa file.
  - Eoln ➔ boolean
  - Eoln(file), file adalah file stream buffer

Eof dan eoln

```
Var i:integer;
begin
 while not eof do
    begin
      write('Masukan bilangan? ');
      read(i);
      writeln('bilangan ', i: 2, '.');
      writeln
    End;
End.
```

Readln(i)

Untuk Berhenti dari loop ini :
windows ➔ Ctrl-Z
Linux ➔Ctrl-D

Error : Run atau Compilation ??

Eof dan eoln

```
var i: integer;
begin
  write('Masukan bilangan? ');
  while not eof do
     begin
          read(i);
          writeln('bilangan ', i: 2, '.');
          writeln;
          write('masukan lagi? ')
     end
end.
```

Read(i)

Untuk Berhenti dari loop ini :
windows ➔ Ctrl-Z
Linux ➔Ctrl-D

Eof dan eoln

- Eoln akan menandai akhir baris dengan karakter kosong, dan men-set nilai eoln=true

Eoln Tidak tepat	Eoln Tepat

read(ch);
if eoln then
selesai dengan 1 baris
else
pemrosesan lainnya
	read(ch);
	if eoln then
	Selesai dengan 1 baris
	else
	begin
	read(ch);
	Pemrosesan lainnya
	End
	{contoh lainn}
	while not eoln do
	get(input); get(input);

I/O dengan file

- Assign(filevar, filename)
  - assign(initfile,'myprog.ini');
- Reset(filevar,filename);
  - reset(initfile,'myprog.ini'); reset(initfile);
- Rewrite(filevar,filename);
  - rewrite(outfile,'myprog.ini');
- Append(filevar,filename);
  - Append(initfile,'myprog.ini'); append(initfile);
- Update(filevar,filename);

I/O dengan file

- Close(filevar);
- flush(filevar);
- Filepos(filevar);
- Filesize(filevar);
- Seek(filevar,n);
- Longfilesize(filevar);
- Longfilepos(filevar);
- Longseek(filevar,n);
- Iochecking(i); i=1 on, i=0 off
- Ioresult(i); i=1 on, i=0 off

Read dan Write
File dan Terminal

- Read /Readln(filevar,variabel)
  - Read, input tanpa di ikuti ganti baris baru
  - Readln, input dengan di ikuti ganti baris baru
- Write/writeln(filevar,variabel)
  - Write, output tanpa di ikuti ganti baris baru
  - Writeln, output dengan di ikuti ganti baris baru
- Explorasi :
  - Keypressed
  - Get_char

Template eoln

```
while not eoln do  {baca satu baris input}
    begin
     read(ch);
     …
     …
    end;
    {sampai disini pasti eoln=true}
  readln;{trap, baca baris berikutnya,}
         {supaya tidak langsung terminate}
```

Template eof

```
while not eof do
   begin
      readln(ch);
      …
   end;

{FPC, crt usesless ??}
{Ctrl-Z ➔windows}
{Ctrl-D ➔unix/linux}
```

Template eoln dan eof

```
while not eof do
 begin
   while not eoln do
    begin
     read(c);
     …
     …
    end;{endofwhile eoln}
   readln;  {baca baris berikutnya}
   …
 end;{endwhile oef}
{FPC, crt usesless ??}
{Ctrl-Z ➔windows}
{Ctrl-D ➔unix/linux}
```

Template Sentinel

```
while not done do
  begin
   count:=0;
   while not eoln do
    begin
     inc(count);
     read(ch);
    end;{endwhile eoln}
    readln;           {baris selanjutnya, eoln}
    if count<>0 then
     begin
       …
       …
     end
    else
     begin
       done:=true; {blank to exit}
     end;
  end;{endwhile done}
```

Array

- Dalam pembahasan ini akan di jelaskan 1 dan 2 di mensi saja
- 1 dimensi ➔vektor,tabel,list
- 2 dimensi ➔matrik
- Fixed size pada run time
- Same type and size
- Akses melalui index

Array 1 dimensi/tabel,vektor,list

- Type
  - Tipedatabaru=array[<ordinal>]of tipedata
  - list=array[maxint]of integer
  - tabel:list;
- Variabel
  - Nama:array[<ordinal>]of tipedata
  - List:array[maxint]of integer
- Akses Array
  - List[1]
  - List[i], list[i+1]

contoh

```
program Arrays;
 var
    a: array[1..5]
 of Integer;
 begin
 end.
```

```
program Arrays;

var
   a: array[1..5] of
Integer;

begin
   a[1] := 12;
   a[2] := 23;
   a[3] := 34;
   a[4] := 45;
   a[5] := 56;
end.
```

contoh

```
program Arrays;

  var
     a: array[1..5] of
  Integer;
     i: Integer;

  begin
     for i := 1 to 5 do
        Readln(a[i]);
  end.
```

```
     a[1] := 23;
     a[2] := 45;
     a[3] := 12;
     a[4] := 56;
     a[5] := 34;
for i := 1 to 4 do
   for j := i + 1 to 5 do
      if a[i] > a[j] then
       begin
         tmp := a[i];
         a[i] := a[j];
         a[j] := tmp;
       end;
```

Studi Kasus 1

- Input :
  - sembarang bilangan integer tidak terurut melalui terminal interaktif , setiap satu input di akhiri dengan menekan tombol enter
  - Simpan input dalam tabel maximal 100 elemen integer
- Output :
  - Cetak bilangan integer yang terurut
  - Cetak nilai max,min, rata-rata, nilai tengah, jumlah bilangan genap, ganjil, bilangan prima

Petunjuk, untuk mengerjakan studi kasus 1-4, gunakan teknik input device melalui interaktif terminal (eof, eoln, read, readln,write,writeln)

Studi Kasus 2

- Input :
  - sembarang string alphabet melalui terminal interaktif
  - Simpan input dalam tabel maximal 100 elemen char
- Output :
  - Cetak total huruf,frekuensi huruf vokal, dan konsonan
  - Cetak prosentase huruf vokal dan konsonan

Misal

input : aku,324- suka= kamu dong-..,

Output : total 15, vokal 7 (46.6%), konsonan 8 (53.4%)

Studi Kasus 3

- Input :
  - sembarang string alphabet melalui terminal interaktif
  - Simpan input dalam tabel maximal 100 elemen char
- Output :
  - Cetak string tersebut dalam urutan terbalik

Misal

input : aku, suka= kamu -..,

Output : ukamakusuka

Studi Kasus 4

- Input :
  - sembarang string alphabet melalui terminal interaktif
  - Simpan input dalam tabel maximal 100 elemen char
- Output :
  - Tentukan apakah string tersebut adalah PALINDROME, cetak YA atau TIDAK , <string> adalah palindrome

Input : a934-=d24/.i-=][g.//a89g\][/.a9*&g-=+_i%$&7d12a
Output :YA, adigagagida adalah palindrome

Array 2 dimensi/matrik

- Type
  - Tipedatabaru=array[<ordinal>, <ordinal>]of tipedata
  - list=array[1..3,1..3]of integer;
  - tabel:list;
- Variabel
  - Nama:array[<ordinal>, <ordinal>]of tipedata
  - List:array[1..3,1..3]of integer;
- Akses Array
  - List[1,1]
  - List[i,j], list[i+1,j+1]

Contoh

```
program Arrays;

var
   r, c: Integer;
   a: array [1..3,1..3] of Integer;

begin
   for r := 1 to 3 do
      for c := 1 to 3 do
         Readln(a[r,c]);
end.
```

Studi Kasus

- Baca dan tulis suatu matrik
  - Satu matrik nxm
  - Dua matrik nxn
- Transpose Matrik
- Untuk 1 matrik nxm tentukan apakah suatu matrik adalah:
  - Simetri ?
  - Satuan ?
  - Sparse ?
  - Berapa jumlah nilai DIAGONAL UTAMA ?

Studi Kasus

- Untuk 2 matrik nxn, tentukan operasi :
  - Penjumlahan, pengurangan, perkalian matrik
  - Apakah matrik tersebut identik (nilai dan dimensi sama)
- Problem : Efisiensi penyimpanan matrik sparse (k,3)??

Prosedur dan Fungsi

- Procedure/Function merupakan sub program, artinya, bila kita membuat program yang terlalu panjang dan ada ekspresi yang di lakukan berulang-ulang , atau kita sering melakukan suatu perhitungan yang sama, lebih baik di program tersebut di pecah menjadi modul atau prosedur atau fungsi.
- Terminologi pemrograman modular
- Paradigma prosedural

Prosedur/Procedure

- Suatu modul atau kumpulan ekspresi atau statemen yang melakukan suatu tugas khusus
- **Procedure nama**(parameter /argumen);
- Nama procedure adalah definisi yang akan dpakai dalam pemanggilan.

```
Procedure Cetak(s:string);
Begin
  Writeln(s);
end
```

Prosedur/Procedure

```
Procedure Cetak(s:string);
Begin
 Writeln(s);
End;
.
.
Cetak('HELO');
```

Nama prosedur
Argumen/parameter
Badan procedure
Procedure call

Prosedur/Procedure

Argumen/Parameter
- Berupa variabel bertipe
- Berjumlah nol atau banyak, di pisahkan dengan ";"

```
Procedure Cetak(s:string);

Str:string='World';
Cetak('HELO');Cetak(str);
```

Formal

Aktual

Prosedur/Procedure

Argumen/Parameter dapat di kirimkan :

Secara Nilai/value

- variabel parameter hanya mengalami perubahan dalam lokal procedure, saat pemanggilan.

Secara Acuan/Referensi

- variabel parameter menalami perubahan nilai baik dalam lokal maupun di luar prosedur, baik saat pemanggilan atau sesudah pemanggilan

Prosedur/Procedure

```
Procedure Tukar1(a:integer;binteger);
Var temp:integer;
Begin
  temp:=a;a:=b;b:=temp;
End;
Procedure Tukar2(var a:integer;
                 var b:integer);
Var temp:integer;
Begin
  temp:=a;a:=b;b:=temp;
End;
```

Variabel lokal prosedur

Fungsi/Function

- Semua propertinya mirip dengan procedure, perbedaanya :
  - Keyword FUNCTION
  - Memberikan nilai hasil balik
  - Seperti fungsi dalam matematika
  - Parameter referensi tidak terpakai

```
Function Tambah(a:integer;b:integer):integer;
Begin
 Tambah:=a+b
End;
```

Fungsi/Function

```
Function Tambah(a:integer;b:integer):integer;
Begin
 Tambah:=a+b
End;
.
.
.
X:integer;
X:=Tambah(2,3);{X:=5}
```

Parameter formal

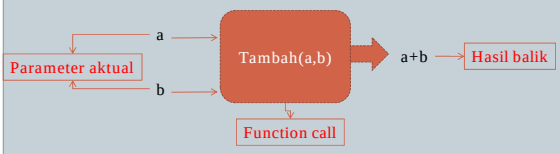
Hasil balik
Harus sesuai spesifikasi fungsi

Parameter aktual

Function call

Fungsi/Function

Representasi Grafik



Fungsi/Function

```
{Buat Fungsi pemangkatan secara iteratif}
Function Pangkat(Y:integer;X:integer):longint;
```

Rekursif

- Suatu fungsi yang memanggil dirinya sendiri, artinya dalam badan fungsi atau prosedur mengandung ekspresi atau pemanggilan terhadap dirinya sendiri
- Model Matematika untuk penyelesaian problem
- Misal : faktorial, binomial, polinomial,deret/barisan, Graph,pengurutan, pencarian, dsb.
- Properti Rekursif :
  - Basis
  - Kondisi Rekursif
  - Ekspresi Rekursif
  - Fungsi Rekursif dengan parameter konvergen ke basis

Rekursif

$$f(n) = \begin{cases} 1, n=0,1 \\ n * f(n-1), n>1 \end{cases}$$

basis

Kondisi rekursif

Ekspresi Rekursif

Fungsi Rekursif

- Basis : Kondisi dasar, trivial solution
- Kondisi Rekursif : Kondisi yang menyebabkan ekspresi rekursif di kerjakan (karena basis tak terpenuhi)
- Ekspresi rekursif : ekspresi yang di dalamnya mengandung fungsi rekursif
- Fungsi rekursif : fungsi yang memanggi l dirinya sendiri dengan parameter menuju ke basis

Rekursif

```
Function Fak(n:integer):longint;
Begin
 if n=1 or n=0 then
   Fak:=1
 Else
   fak:=n*fak(n-1);
End;
```

Problems

1. Input : N,Nilai integer positif
   Output: Faktor N, dan hitung jumlah faktor prima-nya

Misal: input : 10
output: 1,2,5,10 1 prima=2

Problems

2. Input : String paling banyak 255 karakter
   Output: Hitung, jumlah vokal, konsonan, frekuensi vokal (jumlah kemunculan vokal)

Misal :
input : aku adalah lelaki
output: 8,7,a5,u1,e1,i1

Problems

3. Input : Suatu kalimat alfabet,paling banyak 255 karakter (dengan spasi)
   Output: Hitung jumlah kata

Misal: input : aku adalah lelaki
       output: 3 kata

Problems

4. Input : Suatu String paling banyak 255 karakter (alfanumerik dan karakter lainnya)
   Output: hitung karakter alfabet dan karakter selain alfabet

Misal : input :aku-10 di,.=rumah(*&
        output:10,10

Problems

3. Input : Suatu string kalimat alfabet,paling banyak 255 karakter (dengan spasi)
   Output: Geser 2 karakter pertama ke belakang input string

Misal : input : ayo pergi
        output: o pergiay

Problems

4. Input : Suatu String paling banyak 255 karakter (alfanumerik dan karakter lainnya)
   Output: hitung karakter alfabet dan karakter selain alfabet

LAMPIRAN 6. LATIHAN MENULIS PROGRAM

Latihan Menulis Program

Tulislah program kecil di bawah ini, dan analisalah jalannya program serta output yang di hasilkan. Kecepatan menulis dengan benar (identasi), dan jumlah kesalahan (kesalahan sintak, logika dan kompilasi) yang sedikit menentukan seberapa bagus anda menguasai latihan ini.

PROGRAM SEDERHANA

```
program hello;
(* File : HELLO.PAS *)
(* menuliskan Hello ke layar *)
begin
  writeln ( 'hello ') ;
end.

program hellodos;
(* File : HELLODOS.PAS *)
(* menuliskan Hello ke layar *)
uses crt;
begin
  clrscr;
  writeln ( 'hello ') ;
end.
```

INPUT/OUTPUT

```
program baca;
(* File : BACA.PAS *)
(* contoh membaca integer*)
(* kemudian menuliskan nilai yang dibaca *)
var
  a : integer;
begin
  writeln ('Contoh membaca dan menulis, ketik nilai integer: ') ;
  readln (a) ;
  writeln ('nilai yang dibaca : ', a) ;
end.
```

ASSIGNMENT

```
program asign;
(* File : ASIGN.PAS *)
(* Assigntment dan print *)
var
  i : integer;
begin
  writeln ( 'hello' );
  i := 5;
  writeln ( 'Ini nilai i : ',i);
end.
```

```
program asign1;
(* File : ASIGN1.pas *)
(* Assignment dan print *)
var
  i : integer;
  ii : longint;
begin
  writeln ( 'hello');
  i := 1234;
  ii := 123456 ;
  writeln ( 'Ini nilai i=1234 = : ',i);
  writeln ( 'Ini nilai ii=123456 : ',ii);
  writeln ( 'Ini nilai max integer: ',maxint);
  writeln ( 'Ini nilai max longint: ',maxlongint);
end.
```

TIPE DASAR

```
program TDasar;
(* File : TDasar.pas *)
(* Deklarasi, assignment dan penulisan type dasar *)

var
  i     : integer ;
  x,y   : real ;
  found : boolean;
begin
  i:= 5; writeln ('i = ', i);
  x := 0.5; writeln ('x = ', x);
  y := 1.0e + 3; writeln ('y = ', y);
  found:= true; writeln ('Found = ', found);
end.
```

EKSPRESI DAN OPERATOR

```
Program oprator;
(* File : oprator.pas *)
(* Contoh pengoperasian variabel bertype dasar *)

VAR
  Bool1, Bool2, TF : Boolean;
  i, j, hsl : Integer;
  x,y,res : real;
Begin
  writeln ('Utk program ini, baca teksnya dan tambahkan output');
  Bool1 := True; Bool2 := False;
  (* contoh-contoh ekspresi: bukan untuk assignment berulang-ulang *)
  TF := Bool1 And Bool2 ;
  TF := Bool1 or Bool2 ;
  TF := Not Bool1 ;
  TF := Bool1 Xor Bool2 ;
```

```
  (* operasi numerik *)
  i := 5; j := 2 ;
  hsl := i+j; hsl := i - j; hsl := i div j; hsl := i * j;
  hsl := i div j ; (* pembagian bulat *)
  hsl := i Mod j ; (* sisa *)
  (* operasi numerik *)
  x := 5.0 ; y := 2.0 ;
  res := x + y; res := x - y; res := x / y; res := x * y;
  (* operasional relasional numerik *)
  TF := i < j; TF := i > j; TF := i <= j;
  TF := i >= j; TF := i <> y;
  (* operasional relasional numerik *)
  TF := x < y; TF := x > y; TF := x <= y;
  TF := x >= y; TF := x <> y;
end.
```

STRING

```
program manipstr;
(* manipulasi string sederhana *)
var
  str1, str2 : string;
  strr : string;
  stri : string;
  i, kode: integer;
begin
  str1 := 'Saya ';
  str2 := ' Belajar di ITB';
  strr := str1 + str2;
  readln (strr);
  (* koneversi string numerik ke nilai integer *)
  stri := '123';
  val (stri, i, kode ) ; (* amatilah nilai kode setelah eksekusi *)
  writeln(stri, '-', i,'-', kode);
end.

program BACASTR;
(* File : BACASTR.pas *)
(* alokasi string, kemudian mengisinya dengan membaca *)

VAR
  str : string;
  str1 : string;
begin
  writeln ( 'Baca string, maks 256 karakter: ');
  readln ( str);
  writeln ( 'String yang dibaca : ' ,str);
  str1 := str;
  writeln ( 'String yang disalin : ',str1);
end.
```

KONSTANTA

```
program KONSTANTA;
(* File : konstant.pas *)
(* Membaca jari-jari, menghitung luas lingkaran *)
(* Latihan pemakaian konstanta *)

CONST
  pi=3.1415;
VAR
  r : real;
begin
  write ('Jari-jari lingkaran =');
  readln (r) ;
  writeln ('Luas lingkaran = ', pi*r*r);
  writeln (' Akhir program ');
(* Kompilasi, amatilah apa yang terjadi jika komentar sbb. dibuang *)
(* perhatikan option pada kompilator anda *)
(* pi := 10.0; *)
end.
```

```
program KONSTAN2;
(* File : KONSTAN2.PAS *)
(* Menghitung luas lingkaran, dari jari-jari yang dibaca *)
(* Latihan pemakaian konstanta *)
CONST
  pi = 3.1415;
  dua = 2.0;
VAR
  r    : real;
  luas : real;
  kel  : real;
begin
  writeln ( 'Jari-jari lingkaran =');
  readln (r) ;
  luas := pi * r * r;
  writeln ( 'Luas lingkaran = ', luas:6:2);
  kel := dua* pi * r ;
  writeln ('keliling lingkaran= ',luas:6:2);
  writeln ('akhir program ');
end.
```

HIMPUNAN (SET)

```
program Himpunan;
(* File : Himpunan.pas *)
(* Pendefisinian dan pemanfaatan himpunan : SET *)
type
  Hari = (senin,selasa,rabu,kamis,jumat,sabtu,minggu);
  weekday = SET of Hari;
var
  H          : Hari;
  H0, H1, H2 : Hari;
  W          : weekday;
```

```
begin
(* Instruksi berikut salah :type set tidak dapat ditulis/baca*)
(* writeln (' Hari = ', H); *)
(* Assignment : boleh *)
  H1 := selasa;
(* prosedur terdefenisi *)
  H2 := succ (H1);
  H0 := pred (H1);
(* pemanfaatan untk mengontrol pengulangan *)
(* Akan dibahas pada pengulangan *)
  for H := senin to minggu do
  begin
    writeln ( 'Selamat Pagi ...');
    writeln ( 'Ordinal : ', ord (H) );
  end;
(* intruksi CASE : akan dibahas pada analisa kasus*)
  case H1 of
    senin :
      writeln ( 'senin' );
    selasa :
      writeln ( 'selasa' );
    rabu :
      writeln ( 'rabu' );
    kamis :
      writeln ( 'kamis' );
    jumat :
      writeln ( 'jumat' );
    sabtu :
      writeln ( 'sabtu' );
    minggu :
      writeln ( 'minggu' );
    else
      writeln ( 'tidak terdefinisi ');
  end;(*endcase*)
end.
```

ANALISA KASUS, KONDISIONAL

```
(* File : IF1.PAS *)
(* contoh pemakaian IF satu kasus *)
(* membaca nilai integer, menuliskan nilainya jika positif *)
Program IF1;
Var a : integer;
begin
  writeln ( 'Contoh IF satu kasus ');
  write ( 'Ketikkan satu nilai integer : ');
  readln (a);
  if (a >= 0) then
  begin
    writeln ( 'Nilai a positif... ', a);
  end;
end.
```

```
(* File :IF2.PAS *)
(* contoh pemakaian IF dua kasus komplementer *)
(* Membaca sebuah nilai, *)
(* menuliskan 'Nilai a positif , nilai a', jika a >=0 *)
(* 'Nilai a negatif , nilai a', jika a <0 *)
program IF2;
var
  a : integer;
begin
  writeln ('Contoh IF dua kasus ');
  write ( 'Ketikan suatu nilai integer :');
  readln (a);
  if (a >= 0) then
  begin
    writeln ( 'Nilai a positif ', a);
  end else (* a<0 *)
  begin
    writeln ( 'Nilai a negatif ', a);
  end;
end.

(* File : IF3.PAS *)
(* contoh pemakaian IF dua kasus komplementer *)
(* Membaca sebuah nilai, *)
(* menuliskan 'Nilai a positif , nilai a', jika a>0 *)
(* 'Nilai a sama dengan nol , nilai a', jika a =0 *)
(* 'Nilai a negatif , nilai a', jika a <0 *)
program IF3;
var
  a : integer;
begin
  writeln ( 'Contoh IF tiga kasus');
  write ( 'Ketikkan suatu nilai integer :');
  readln (a);
  if (a > 0) then
  begin
    writeln ( 'Nilai a positif ', a);
  end else
    if (a=0) then
    begin
      writeln ( 'Nilai a sama dengan nol ', a);
    end else if (a<0) then
      begin
        writeln ( 'Nilai a negatif ', a);
      end;
end.
```

```
program KASUS;
(* File : KASUS.PAS *)
(* Contoh kasus dengan intruksi CASE *)
VAR
  cc : char;
begin
  writeln ( 'Ketikkan sebuah huruf, akhiri dengan RETURN ');
  readln (cc);
  case cc of
    'a' : begin
            writeln ( ' Yang anda ketik adalah a ' );
          end;
    'u' : begin
            writeln ( ' Yang anda ketik adalah u ' );
          end;
    'e' : begin
            writeln ( ' Yang anda ketik adalah e ' );
          end;
    'o' : begin
            writeln ( ' Yang anda ketik adalah o ' );
          end;
    'i' : begin
            writeln ( ' Yang anda ketik adalah i ' );
          end
    else writeln ( ' Yang anda ketik adalah huruf mati atau angka' );
  end;(*endcase*)
end.

(* File : wujudair.PAS *)
(* contoh pemakaian IF tiga kasus : wujud air *)
program wujudair;
var T: integer;
begin
  writeln ( 'Contoh IF tiga kasus ' );
  write ( 'Temperatur (der. C) = ' );readln (T);
  if (T < 0) then
  begin
    writeln ( 'Wujud air beku ');
  end else
    begin
      if ( (0<=T) and (T<=100) ) then
      begin
        writeln ( 'Wujud air cair ');
      end else
        begin
          if (T>100) then
          begin
            writeln ( 'Wuju air uap/gas ');
          end;
        end;
    end;
end.
```

```
program MAX2;
(* File :MAX2.PAS *)
(* Maksimum dua bilangan yang dibaca *)
VAR
  a,b : integer;
begin
  writeln ( 'Maksimum dua bilangan : ' );
  write ( 'Ketikan bilangan pertama : ' );
  readln (a) ;
  write ( 'Ketikan bilangan kedua : ' );
  readln (a) ;
  if (a >= b) then
  begin
    writeln ( 'Nilai a yang maksimum ', a);
  end
  else
  begin
    writeln ( 'Nilai b yang maksimum ', b) ;
  end ;
end.
```

SUBPROGRAM

```
program subprg;
(* File : subprg.PAS *)
(* contoh pendefinisian dan pemanggilan FUNGSI dan PROSEDUR *)
(* deklarasi dan badan procedure/fungsi LOKAL *)
(* Konsep yang harus dijelaskan :*)
(* - perbedaan fungsi dan prosedur *)
(* - parameter formal dan aktual *)
(* - passing parameter by value dan by ref *)
var
  a, b : integer;
function maxab (a,b : integer) : integer;
begin (* mencari maksimum dua bilangan bulat *)
    if a>=b then
    begin
      maxab := a;
    end else
    begin
      maxab := b;
    end;
end;

procedure tukar (var a, b : integer);
(* menukar nilai dua buah variabel a dan b *)
(* parameter input/output *)
Var temp : integer;
begin (* menukar dua bilangan bulat *)
  temp := a;
  a := b;
  b := temp;
end;
```

```
begin (* program utama *)
(* Membaca dua bilangan integer *)
(* Menuliskan maksimum dua bilangan yang dibaca dg memanggil fungsi*)
(* Menukar kedua bilangan dengan 'prosedur' *)
writeln ( 'Maksimum dua bilangan : ');
  writeln ( 'Ketikkan bilangan pertama : ');
  readln (a) ;
  writeln ( 'Ketikkan bilangan kedua : ');
  readln (b) ;
  writeln ( 'Ke dua bilangan : a = ',a) ;
  writeln ( ' b = ',b) ;
  writeln ( 'Maksimum = ', maxab (a,b) ) ;
  writeln ( 'Tukar kedua bilangan... ') ;
  tukar ( a, b ) ;
  writeln ( 'Ke dua bilangan setelah tukar: a = ',a) ;
  writeln ( ' b = ',b) ;
end.
```

LINGKUP (SCOPE)

```
program Lingkup;
(* File : Lingkup.pas *)
(* arti nama : Lingkup dan masa hidup variabel; parameter by value,
   by ref *)
Var a, b : integer;
procedure Plusl (var x:integer);
(* x adalah parameter input/output *)
(* Menambah nilai thd Prosedur Plusl *)
(* Fungsi lokal thd Prosedur Plus1 *)
  function Incr (i:integer) : integer;
  (* mengirimkan nilai i ditambah 1 *)
  begin
    Incr := i+1;
  end;
begin
  x := Incr(x) ; (* nilai parameter input x ditambah 1 *)
end;

Procedure Swap (var a,b : integer );
(* perhatikan bada a dan b disini dengan deklarasi a dan b global *)
Var temp : integer; (*variabel lokal *)
begin
  temp := a;
  a:= b;
  b:= temp;
end;

function Plus2 (i:integer) : integer;
(* Mengirimkan nilai i ditambah 2 *)
begin
  Plus2 := i+2;
end;
```

```
begin
  (* berikut tidak dikenal *)
  (* writeln ("Nilai i= ',i); *)
  a := 2;
  (* Berikut ini salah *)
  (* b := Incr (a) ; *)
  b := a;
  Plus2 (a);
  swap (a,b);
  (* Berapa hasilnya ?? *)
  writeln ('1. a= ',a, 'b= ', b );
  b := Plus2 (b);
  (* Berapa hasilnya ?? *)
  writeln ('2. a= ',a, 'b= ', b );
end.
```

```
program FuncRec;
(* File : FuncRec.pas *)
(* Fungsi yang harus mengembalikan type bentukan : tidak mungkin *)
type
  Point = record
    x : integer; (* absis*)
    y : integer; (* ordinat*)
  end;
var T, T1, T2 : Point;

procedure Tulis (T:Point);  (* menuliskan sebuah titik T *)
begin
  writeln ( 'Titik T ( ',T.x, ', ',T.y, ') ');
end;

(* function MidPoint (T1,T2: Point) : Point; *)
(* Menghasilkan Titik tengah T1,T2 berupa titik *)
(* karena fungsi dalam bahasa Pascal tidak bisa mengembalikan *)
(* type record *)
(* berikut ini transformasi untuk mendapatkan efek yang dimaksud *)
procedure TTkTengah (T1,T2 : Point; var MidPoint : Point) ;
(* titik hasil *)
(* Menerima T1 dan T2 dua buah Point *)
(* Menghasilakan Midpoint : sebuah VARIABEL bertype Point *)
begin
  MidPoint.x := ((T1.x + T2.x) div 2 );
  MidPoint.y := ((T1.y + T2.y) div 2 );
end;

begin
  T1.x := 0; T1.y := 0;
  T2.x := 10; T2.y := 10;
  Tulis (T1);Tulis (T2);
  TtkTengah (T1,T2,T);  Tulis (T);
end.
```

PENGULANGAN

```
program PRIFOR;
(* File : PRIFOR.PAS *)
(* Baca N, Print 1 s/d N dengan FOR *)
var
  i : integer;
  N: integer;
begin
  writeln ( 'Baca N, print 1 s/d N ');
  write ( 'N = ');
  readln (N) ;
  for i:=1 to N do
  begin
    writeln (i);
  end; (* FOR *)
  writeln ( 'Akhir program ');
end.
```

```
program PRIW;
(* File : PRIW.PAS *)
(* Baca N, *)
(* Print i = 1 s/d N dengan WHILE *)
VAR
  n : integer;
  i : integer;
begin
  write ( 'Nilai N = ');
  readln (N);
  i := 1 ;
  writeln ( 'Print i dengan WHILE: ');
  while (i<=N) do
  begin
    writeln (i);
    i := i + 1;
  end ; (* i>N *)
end.
```

```
program PRIREP;
(* File : PRIREP.PAS *)
(* contoh baca N, print 1 s/d n dengan REPEAT *)
Var N : integer; i : integer;
begin (* Program *)
  write ( 'Nilai N= '); readln (N);
  i := 1 ;
  writeln ( 'Print i dengan REPEAT: ');
  repeat
    writeln (i);
    i := 1;
  until (i > N);
end.
```

```
program PRITER;
(* File : PRITER. Pas *)
(* Baca N, *)
(* Print i = 1 s/d N dengan ITERATE *)
VAR
  N : integer;
  i : integer;
  stop : boolean;
begin
  write ( 'Nilai N = ');
  readln (N);
  i := 1;
  writeln ( 'Print i dengan ITERATE : ');
  stop := false;
  repeat
    writeln (i);
    if (i=N) then stop := true else
    begin i := i + 1; end;
  until stop; (* i= N *)
end.

program KASUSREP;
(* File : KASUSREP.PAS *)
(* Contoh kasus dengan switch dan pengulangan *)
(* membaca karakter sampai user mengetikkan q *)
VAR
  cc : char;
  quit : boolean;
begin
  repeat
    quit := false;
    write (' Ketikkan sebuah huruf, akhiri dengan q :');
    readln (cc);
    case cc of
      'a' : begin
            writeln (' Yang anda ketik adalah a ');
            end;
      'u' : begin
            writeln (' Yang anda ketik adalah u ');
            end;
      'e' : begin
            writeln (' Yang anda ketik adalah e ');
            end;
      'i' : begin
            writeln (' Yang anda ketik adalah i ');
          end;
      'q' : begin
            quit := true;
          end;
      else writeln (' Yang anda ketik adalah huruf mati ');
    end; (* case *)
  until (quit);
```

```
  writeln (' Akhir program... sebab anda mengetik q ');
end.
```

TABEL (ARRAY)

```
program TABEL;
(* File : TABEL.PAS *)
(* latihan array : mengisi dg assignment, menulis *)
VAR
  i : integer;
  tab : array [1..10] of integer;
  N : integer;
begin
  N := 5;
  writeln ( 'Isi dan print tabel: ');
  (* isi dengan assignment *)
  for i := 1 to N do
  begin
    tab [i] := i;
    tab [i] := i;
  end;
(* traversal : print *)
  for i := 1 to N do
  begin
    writeln ( 'i= ',i, ' tab[i]= ', tab[i] );
  end;
end.
```

TABEL MULTI DIMENSI

```
program Tab2dim;
(* File : Tab2dim.pas *)
(* Tabel integer dua dimensi (matriks) *)
type
  (* Cara I : sebagai array dua dimensi *)
  MatInt = array [1..3, 1..3] of integer;
  (* Cara 2 : sebagai array of array *)
  MatArr = array [1..3] of array [1..3] of integer;
var
  M1 : MatInt;
  MA1 : MatArr;
  i , j : Integer ;
begin
  writeln (' Array dua dimensi : ');
  (* Mengisi Matrik dua dimensi *)
  for i := 1 to 3 do
  begin
    for j := 1 to 3 do
    begin
      M1 [i,j] := i * j;
    end; (* for j *)
  end;
```

```
  (* Menulis hasil isian di atas *)
  for i := 1 to 3 do
  begin
    for j:= 1 to 3 do
    begin
      write (' (i, j) = ', i, ',', j, ' => M1 [i, j] = ', M1 [i, j]);
    end;
    writeln;
  end;
  writeln ( ' Array of array : ');
  (* Mengisi array of array : perhatikan cara mengacu elemen *)
  for i := 1 to 3 do
  begin
    for j := 1 to 3 do
    begin
      MA1 [i] [j] := i*j;
    end;
  end;
  (* Menulis hasil isian di atas *)
  for i := 1 to 3 do
  begin
    for j := 1 to 3 do
    begin
    (* Cobalah dua instruksi write sbb : *)
    (* write(' (i,j) = ', i, ', ', j, ' => MA1[i,j] = ',MA1[i,j] ); *)
      write ('(i,j) = ', i, ', ', j, ', => MA1 [i,j] = ',MA1 [i][j] );
    end;
    writeln;
  end;
end.
```

LAMPIRAN 7. CONTOH SOAL DAN PEMBAHASAN

Contoh Soal dan Pembahasan
Ujian Praktek Pemrograman Pascal

Contoh 1.

Cetak Angka *(Soal ini pernah diberikan pada Practice Session OSN 2003, Balikpapan)*

Ketentuan umum :

Nama Program : CETAK.PAS
Batas *Run-time* : 1 detik / test case
Nama File Masukan : CETAK.IN
Nama File Keluaran : CETAK.OUT

Tulis sebuah program yang membaca sebuah bilangan bulat *n* dan mencetak bilangan bulat dari 1 sampai dengan n^2.

FORMAT MASUKAN (Nama File: CETAK.IN)
Masukan terdiri dari sebuah bilangan bulat *n* ($1 = n = 10$).

CONTOH MASUKAN

```
3
```

FORMAT KELUARAN (Nama File: CETAK.OUT**)**
Keluaran terdiri dari *n* baris. Baris ke-*i* ($1 = i = n$) berisi ($2i - 1$) bilangan bulat, dengan tiap bilangan bulat dipisahkan oleh sebuah spasi. Setiap bilangan bulat terurut menurun sehingga setiap bilangan bulat yang tercetak selalu lebih besar daripada bilangan bulat di sebelah kanannya (bila ada) dan selalu lebih besar daripada bilangan-bilangan bulat pada baris-baris di bawahnya (bila ada).

CONTOH KELUARAN

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

PEMBAHASAN DAN SOLUSI

Oleh : Ilham Winata Kurnia, Kontestan IOI 2002 Yong-In, Korea

Dalam soal ini, kita diminta untuk menampilkan bilangan 1 sampai dengan N^2 secara terurut menurun. Selain itu, setiap barisnya, kita perlu menampilkan ejumlah bilangan (bila diperhatikan, pada baris ke-i, ada 2 * i - 1 bilangan yang ditampilkan, dan ini adalah barisan bilangan ganjil). Jadi, kita cukup elakukan iterasi hingga ada tepat 2 * i - 1 bilangan yang tercetak pada baris ke-i (dengan menggunakan perintah write()). Setelah itu baru kita cetak pengganti barisnya (dengan menggunakan perintah writeln()).

Solusi :

```
program CetakAngka(input, output);

Const InFile = 'CETAK.IN';
      OutFile = 'CETAK.OUT';
      MAXN = 10;

var i, j, k, n : integer;

begin
assign(input, InFile);
assign(output, OutFile);
reset(input);
rewrite(output);

readln(n);
j := n * n; k := j;
for i := 1 to n do
  begin
  write(j); dec(j);
  while k - j >= (2 * i) - 1 do begin write(' ', j); dec(j); end;
  writeln;
  end;

close(input);
close(output);
end.
```

Contoh 2

Menghitung Perulangan

Ketentuan umum :

Nama Program : HITUNG.PAS
Batas *Run-time* : 1 detik / test case
Nama File Masukan : HITUNG.IN
Nama File Keluaran : HITUNG.OUT

Diberikan dua buah untaian huruf (*string*), hitung berapa kali string kedua muncul sebagai bagian dari string pertama. Asumsikan bahwa tiap kemunculan dari string kedua pada string pertama boleh saling menimpa (*overlap*). Panjang string pertama maksimal 10000, sementara panjang string kedua maksimal 200. String didefinisikan sebagai untaian karakter-karakter dengan kode ASCII 32 – 127 yang dibatasi oleh karakter-karakter dengan kode ASCII yang tidak termasuk dalam jangkauan 32– 127 tersebut.

FORMAT MASUKAN (Nama File: HITUNG.IN**)**
Masukan terdiri dari dua baris. Baris pertama berisi string pertama, sementara baris kedua berisi string kedua.

CONTOH MASUKAN

```
abcdefabcghiabcabcjklmnlabcw
abc
```

FORMAT KELUARAN (Nama File: HITUNG.OUT**)**
Keluaran hanya terdiri dari sebuah baris berisi sebuah bilangan bulat yang menyatakan banyak kemunculan string kedua pada string pertama.

CONTOH KELUARAN

```
5
```

PEMBAHASAN
Oleh : Ilham Winata Kurnia, Kontestan IOI 2002, Yong In – Korea

Inti dari "Menghitung Perulangan" tidak lain adalah berulang kali menyimulasikan perintah *find*, yang kita sering temui pada *software-software word processing* seperti Notepad, Edit, Star Office, dan sebagainya. Ada beberapa cara yang dapat digunakan untuk melakukan hal ini. Akan tetapi, untuk tingkat ini, hanya satu cara yang akan dibahas di sini, yaitu dengan *Brute Force*.

Permasalahan ini bisa dibagi menjadi 2 bagian: membaca input dan mencari keberadaan substring. Ada satu hal yang menyebabkan membaca input menjadi bagian tersendiri, yaitu panjang string pertama melebihi 255. Oleh sebab itu, kita tidak dapat serta merta melakukan pembacaan dengan menggunakan readln (walaupun nanti kita akan lihat sebuah pengecualian).

Salah satu cara untuk mengatasi hal ini adalah dengan mendefinisikan sebuah tipe variabel sendiri yang berupa array[1..10000] of char. Untuk membaca sebuah baris pada masukan, maka kita baca karakter demi karakter sampai kita temui karakter penanda akhir baris. Di Linux, karakter penanda akhir baris adalah ASCII #10 alias *newline character*, sementara pada Windows, karakter penanda akhir baris adalah rentetan ASCII #13 atau *linefeed character* dan ASCII #10.

Seperti yang dikatakan di atas, ada beberapa cara yang dapat dilakukan untuk menentukan keberadaan substring pada sebuah string, tapi kita akan hanya membahas satu saja. Cara *brute force* mengiterasi semua karakter pada string dan membandingkan setiap karakter pada bagian string yang sedang diiterasi dengan karakter-karakter dari substring. Bila setiap perbandingannya adalah benar, maka substring tersebut ada bagian dari string. Kita hanya cukup menghitung berapa kali perbandingan tersebut benar untuk mengetahui berapa kali kemunculan substring pada string tersebut.

SOLUSI

```
program MenghitungPerulangan(input, output);

Const InFile = 'HITUNG.IN';
      OutFile = 'HITUNG.OUT';
      MAXA = 10000;
      MAXB = 200;

var a : array[1..MAXA + 1] of char;
    b : array[1..MAXB + 1] of char;
    cta, ctb, ans : integer;
    i, j : integer;
    cek : boolean;
```

```
begin
assign(input, InFile);
assign(output, OutFile);
reset(input);
rewrite(output);

cta := 0; ctb := 1; ans := 0;

while not eoln do
  begin
  inc(cta);
  read(a[cta]);
  end;

{ read input }
b[ctb] := #13;
while (b[ctb] = #10) or (b[ctb] = #13) do read(b[ctb]);
while not eoln and not eof do
  begin
  inc(ctb);
  read(b[ctb]);
  end;

{ find substring locations }
for i := 1 to cta - ctb + 1 do
  begin
  cek := true;
  for j := 1 to ctb do
    if a[i + j - 1] <> b[j] then begin cek := false; break; end;
  if cek then inc(ans);
  end;

writeln(ans);

close(input);
close(output);
end.
```

Adapun solusi lain yang lebih mudah adalah untuk menggunakan tipe ANSIString yang diberikan oleh Free Pascal. Dengan menggunakan tipe ini, kita tinggal menggunakan perintah readln, writeln, copy, dan pos. Hal ini dimungkinkan karena ANSIString dapat menyimpan untaian karakter dengan jumlah karakter hampir tak terhingga, tapi tetap mempertahankan sifat-sifat yang dimiliki oleh tipe variabel string. Dengan demikian, kita cukup melakukan satu iterasi saja, yaitu iterasi untuk mencari posisi dimana ada kemunculan substring pada string. Walaupun demikian, waktu yang dibutuhkan oleh program ini lebih kurang sama dengan program di atas karena perintah pos sendiri melakukan iterasi untuk mencari posisi substring. Di bawah ini adalah contoh source codenya:

```
program MenghitungPerulanganAlternatif(input, output);

var a, b : ANSIString;
    la, i, j, ans : integer;

begin
assign(input, 'HITUNG.IN');
assign(output, 'HITUNG.OUT');
reset(input);
rewrite(output);

readln(a);
readln(b);
la := length(a);
i := 1; j := 1; ans := 0;

{ cari substring sebanyak-banyaknya }
while j <> 0 do
  begin
  j := pos(b, copy(a, i, la));
  i := i + j;
  if j <> 0 then inc(ans);
  end;

writeln(ans);

close(input);
close(output);
end.
```

Seperti halnya soal "Cari Maksimum", ada 10 test case yang digunakan. Berikut rinciannya:

No.	Panjang String 1	Panjang String 2	Keterangan
1	1	1	Tidak ada kemunculan
2	1	2	Tidak ada kemunculan
3	495	1	Mengetes pembacaan input, dengan string 2 adalah sebuah spasi
4	10000	2	String 1 berisi hanya 3 jenis karakter
5	100	3	String 1 berisi hanya 3 jenis karakter
6	500	5	String 1 berisi hanya 2 jenis karakter, sementara string 2 berisi hanya sebuah jenis karakter
7	1520	20	String 1 berisi hanya 4 jenis karakter random dan disisipkan sebuah kemunculan dari string 2
8	10000	200	Mengetes kasus terbesar. String 1 dan 2 hanya terdiri dari sebuah jenis karakter
9	4517	40	Sama seperti 6, tapi string 2 berisi 2 buah jenis karakter
10	8188	100	Sama seperti 1, tapi string 2 juga berisi 3 jenis karakter

Soal 3.

Tempat Tidur Bebek

Nama File : TIDUR.PAS

Pak Dengklek punya N (1 <= N <= 2500) bebek yang tidur di sebuah kandang besar dengan K kamar yang dinomori 0 sampai dengan K-1. Bebek ke-i dinomori secara unik dengan nomor S_i (1 <= S_i <= 1000000). Setiap bebek tahu di mana untuk tidur karena dia tidur di kamar nomor S_i mod K. Tentu saja, para bebek tidak mau membagi tempatnya untuk tidur.

Diberikan sebuah himpunan bebek dan nomornya, tentukan nilai minimum K sedemikian sehingga tidak ada 2 bebek yang tidur di kamar yang sama.

FORMAT MASUKAN (TIDUR.IN)

Baris 1 : Sebuah bilangan bulat untuk N
Baris 2..N+1 : Sebuah bilangan bulat yang menyatakan nomor bebek

CONTOH MASUKAN

```
5
4
6
9
10
13
```

FORMAT KELUARAN (TIDUR.OUT)

Sebuah baris dengan nilai minimum K pada baris tersebut.

CONTOH KELUARAN

```
8
```

PEMBAHASAN

Oleh : Ilham Winata Kurnia, Anggota TOKI 2002, Yong-In - Korea Selatan

Soal ini merupakan terjemahan dari soal berjudul "Cows in Beds" yang pernah dikeluarkan oleh United States of America Computing Olympiad (USACO) dalam salah satu kontesnya. Berikut adalah solusi dari salah satu pesertanya, Bruce Merry, yang berhasil melewati tes data yang dibuat oleh USACO (tes datanya lebih menantang daripada yang digunakan untuk seleksi TOKI).

Cara yang paling terlihat untuk menyelesaikan masalah ini adalah untuk melakukan loop melalui semua kemungkinan nilai K dan pada setiap iterasi, kita cari di mana setiap bebek tidur sambil menggunakan sebuah *array of boolean* untuk mengecek adanya perulangan (yaitu, ada dua bebek dalam sebuah kamar). Untuk mempercepat, kita dapat memulai iterasi dari N karena tidak mungkin kita dapat menempatkan N bebek di kurang dari N kamar tanpa adanya yang membagi tempat tidurnya (tampak jelas, tapi para matematikawan berpikir bahwa ini cukup penting untuk disebut *pigeon hole principle* atau prinsip burung dara).

Cara yang lebih ambisius adalah untuk menandai (dengan menggunakan sebuah *array of boolean*) semua nilai dari K yang tidak dapat digunakan dan carilah nilai pertama yang benar. Nilai-nilai K yang tidak dapat digunakan adalah faktor dari $|S_i-S_j|$ untuk semua nilai (i, j). Sayangnya, ada banyak sekali pasangan (i, j), dan mencari faktor-faktor adalah sangat lambat. Mungkin kita bisa membuat beberapa trik yang memanfaatkan fakta bahwa ada **banyak** faktor yang diulang, bahkan nilai selisih yang berulang. Namun, dengan batas memori 16MB kita akan kesulitan untuk menyeimbangkan antara batas waktu dan memori, dan selain itu kita masih melakukan banyak sekali pembagian.

Pendekatan yang Bruce Merry gunakan adalah untuk menandai $|S_i-S_j|$ sebagai nilai K yang tidak dapat digunakan, untuk setiap pasang (i, j), lalu menggunakan algoritma yang pertama tapi dengan melewati nilai-nilai tersebut. Karena nilai dari $|S_i-S_j|$ adalah dalam rentang 0..1000000, maka memori bukan suatu masalah. Selain itu, karena tidak ada operasi pembagian atau perkalian, maka cara ini relatif cepat. Ide dasarnya adalah bahwa untuk tes data yang besar (tentu saja, tes-tes ini adalah tujuan dari optimasi kita) kita perlu menghilangkan sebanyak mungkin nilai K sehingga mengecek nilai-nilai yang tersisa dengan cara pertama adalah jauh lebih cepat.

Salah satu optimasi lain adalah untuk juga menandai nilai $|S_i-S_j|/2$ dimana $|S_i-S_j|$ adalah bilangan genap. Ini menjamin bahwa semua nilai K antara 333334 dan 1000000 ditandai karena semua faktor lain dari $|S_i-S_j|$

harus paling besar 1000000 / 3=333333. Dengan katalain, begitu nilai K > 333333, maka kita cukup melakukan sebuah tes penuh pada nilai K berikutnya yang tidak ditandai, karena nilai tersebut adalah jawabannya.

SOURCE CODE (Oleh : Bruce Merry)

```
program cows_in_bed;
const
     inname = 'TIDUR.IN';
     outname = 'TIDUR.OUT';
var
   N : integer;
   K : longint;
   Si : array[1..5000] of longint;
   used : array[0..999999] of boolean;
   nogood : array[1..1000000] of boolean;
procedure readin;
var
   f : text;
   i : integer;
begin
     assign(f, inname);
     reset(f);
     readln(f, N);
     for i := 1 to N do
      readln(f, Si[i]);
     close(f);
end;
procedure makenogood;
var
   i, j : integer;
begin
     fillchar(nogood, sizeof(nogood), 0);
     for i := 1 to N - 1 do
      for j := i + 1 to N do
       nogood[abs(Si[i] - Si[j])] := true;
end;

procedure solve;
var
   i : integer;
   cur : longint;
   flag : boolean;
begin
     K := N - 1;
     flag := false;
     repeat
      inc(K);
      if nogood[K] then continue;
      flag := true;
      fillchar(used, sizeof(used[0]) * K, 0);
      for i := 1 to N do
```

```
       begin
        cur := Si[i] mod K;
        if used[cur] then
         begin
          flag := false;
          break;
         end;
        used[cur] := true;
       end;
     until flag;
end;
procedure writeout;
var
   f : text;
begin
     assign(f, outname);
     rewrite(f);
     writeln(f, K);
     close(f);
end;
begin
     readin;
     makenogood;
     solve;
     writeout;
end.
```

Contoh 4.

Spiral Huruf

Huruf-huruf yang disusun dalam konfigurasi di bawah ini adalah huruf alphabetis biasa. Tulislah program untuk mencetak spiral huruf dengan ukuran N (1 = N = 5, dan N ganjil) searah jarum jam. Spiral bisa dibuat dari tengah, atau dari salah satu pojok (tergantung dari inputnya). Data masukan adalah N dan A. A = 1 artinya tengah, A = 2 artinya pojok kiri atas, A = 3 artinya pojok kanan atas, A = 4 artinya pojok kanan bawah, A = 5 artinya pojok kiri bawah.

FORMAT MASUKAN
Data input disimpan dalam text file SPIRAL.IN. File ini berisi nilai N dan A.

FORMAT KELUARAN
Simpan hasil tampilan spiral huruf pada file SPIRAL.OUT.

CONTOH MASUKAN

```
3 4
```

CONTOH KELUARAN

```
E F G
D I H
C B A
```

PEMBAHASAN

Dalam masalah ini, kita diinginkan untuk membuat sebuah program yang dapat menghasilkan sebuah spiral huruf dalam bentuk mirip matriks N x N dengan cara yang sudah ditentukan (A). Pengertian spiral huruf dapat langsung dimengerti dengan melihat contoh keluaran. Ada sebuah frase kunci yang amat penting untuk menyelesaikan masalah ini, yaitu “searah jarum jam”.

Sebetulnya, soal ini tidak jelas karena tidak memberikan keterangan bagaimana spiral harus dibentuk bila A = 1. Ada 4 kemungkinan, yaitu bergerak ke atas, kiri, bawah dan kanan setelah mengisi kotak paling tengah.

Oleh karena itu, mari kita asumsikan bahwa setelah mengisi kotak yang ditengah maka kita bergerak ke sebelah kanan. Misal, untuk input 3 1, maka outputnya akan menjadi

```
G H I
F A B
E D C
```

SOLUSI 1

Bila kita melihat batas N, maka kita dapat simpulkan bahwa N hanya dapat bernilai 1, 3 dan 5. Karena untuk setiap N hanya ada 5 kemungkinan nilai A, maka total hanya ada 15 kemungkinan input. Mengingat jumlah yang begitu kecil ini, maka kita bisa mengerjakan semuanya secara manual, kemudian memasukkannya ke dalam program secara manual. Dengan menggunakan perintah if-then-else, langsung cetak solusinya yang telah kita masukkan secara manual ke dalam program.

SOLUSI 2

Cara yang lain untuk menyelesaikan masalah ini adalah untuk membuat program yang menyimulasikan pergerakan posisi huruf. Bayangkanlah output itu terletak pada suatu system koordinat kartesius. Posisi pojok kiri bawah memiliki koordinat (1, 1) dan posisi pojok kanan atas memiliki koordinat (N, N). Dalam contoh di atas, C berkoordinat (1, 1) dan G (3, 3). Kemudian, setiap huruf disimpan dalam sebuah tabel 2 dimensi dengan besar N x N.

Kita misalkan koordinat-x dengan variabel *x* dan koordinat-y dengan variabel y. Bila kita ingin bergerak ke atas, maka *x* kita tambah dengan 1; turun, *x* kita kurang den gan 1; kanan, *y* kita tambah dengan 1; kiri, *y* kita kurang dengan 1. Karena dalam soal dikehendaki untuk mencetak spiralnya searah jarum jam, maka marilah kita buat aturan arah gerak sebagai berikut: kanan-bawah-kiri-atas-kanan-…. Yang sekarang jadi masalah adalah kapan kita harus berubah arah gerak.

Dengan mencoba-coba, maka kita dapat aturan sebagai berikut:

- Bila A = 1 maka:

```
(x, y) = koordinat tengah tabel
Tabel[x, y] = “A”
Arah gerak = atas
```

```
BILA masih ada kotak kosong LAKUKAN
      JIKA bisa bergerak dengan Arah gerak berikutnya MAKA
            ubah Arah gerak menjadi Arah gerak berikutnya
      JIKA TIDAK MAKA
            bergerak sesuai dengan Arah gerak yang sekarang
            Tabel[x, y] = huruf selanjutnya
```

- Bila A = 2 maka:

```
(x, y) = koordinat pojok kiri atas
Tabel[x, y] = "A"
Arah gerak = kanan

BILA masih ada kotak kosong LAKUKAN
      JIKA bisa bergerak dengan Arah gerak saat ini MAKA
            bergerak sesuai dengan Arah gerak saat ini
            Tabel[x, y] = huruf selanjutnya
      JIKA TIDAK MAKA
            Arah gerak = Arah gerak berikutnya
```

- Langkah penyelesaian untuk A = 2 berlaku juga untuk A = 3, A = 4, dan A = 5. Hanya saja, posisi awal (x, y) diubah sesuai dengan yang tertulis di soal dan arah gerak pertama kali juga disesuaikan. Misal, untuk A = 5, maka posisi awal (x, y) adalah koordinat pojok kiri bawah dan Arah gerak = atas.

Contoh 5.

Pembagian Panjang

Untuk setiap integer [bilangan bulat] yang diberikan, cetaklah hasil pembagian integer jika sebuah integer masukan dibagi dengan 13. Sisa pembagian tidak perlu di tulis. Jika hasil pembagian adalah 0, maka tuliskan saja nol sebagai outputnya.

FORMAT MASUKAN

Terdiri atas sejumlah baris integer yang minimal terdiri atas 1 digit dan paling panjang 50 digit tanpa spasi sebelum dan sesudah integer di setiap barisnya.

FORMAT KELUARAN

Hasil pembagian integer dari data input dengan 13.

CONTOH

MASUKAN	KELUARAN
0	0
12	0
13	1
14	1
25	1
26	2
262626	20202
1313131313131313131313131313	101010101010101010101010101

PEMBAHASAN

Apa yang dituntut dari soal ini cukup jelas yaitu untuk mencetak hasil bulat dari suatu bilangan bulat yang dibagi dengan 13. Akan tetapi ada dua hal yang menjadi masalah. Pertama, panjang bilangan bulat bisa mencapai 50 digit. Kedua, tidak tertulis apakah input selalu merupakan bilangan bulat positif atau bisa juga

merupakan bilangan bulat negatif. Untuk masalah yang kedua, amat mudah untuk dipecahkan dengan 1 perintah if. Akan tetapi, untuk menyelesaikan masalah yang pertama butuh lebih dari itu. Berikut ini kita asumsikan bahwa input telah diubah ke bilangan bulat positif.

Dengan adanya petunjuk bahwa panjang input bisa mencapai 50 digit, maka kita tidak bisa menyimpan input dalam suatu variabel bilangan biasa. Longint hanya bisa menampung maksimum 10 digit bilangan bulat. Comp (dalam Turbo Pascal) atau Int64 (dalam Free Pascal) hanya sanggup menampung 19 digit bilangan bulat. Mungkin ada yang beranggapan bahwa variabel-variabel *floating point* seperti Real (dalam Pascal) atau Double dapat menampung bilangan bulat tersebut. Hal itu memang benar, tapi yang disimpan hanya bisa sampai 23 digit pertama beserta tingkat eksponennya dalam basis 10 (sistem desimal). Bila pembagian dilakukan, maka hasilnya hanya akurat sampai digit ke 22. Dengan demikian, kita harus memikirkan suatu teknik penyimpanan dengan menggunakan tipe variabel dasar yang tersedia.

Adalah suatu yang alami bila kita berpikir untuk menggunakan array. Misalkan input kita simpan dalam array dari integer dengan panjang 50. Setiap digit dari input kita simpan dalam 1 elemen. Sekarang timbul masalah baru, yaitu bagaimana cara mendapatkan setiap digit dari input. Ada beberapa cara. Salah satunya adalah untuk membaca tiap baris sebagai sebuah string. Setelah itu, setiap karakter dari string tersebut dikonversi ke nilai digit menggunakan nilai ASCII dari karakter tersebut. Kemudian, panjang string itu menjadi panjang bilangan.

Setelah kita mendapatkan input tersebut dalam array, maka kita tinggal melakukan pembagian panjang dengan cara sesuai dengan yang telah diajarkan sewaktu kita masih duduk di sekolah dasar. Berikut ini kode semu *(pseudocode)* langkah-langkah untuk memecahkan soal ini. Sekedar catatan: "\" adalah pembagian yang hanya menghasilkan bilangan bulatnya saja.

```
Bil = jawab = array dari integer dengan panjang 50.
Pan Bil = Panjang dari Bil
Pan jawab = Panjang dari jawab = 0
Ubah input ke dalam array Bil
Sisa = 0

DARI I = 1 SAMPAI DENGAN Pan Bil LAKUKAN
      Sisa = Sisa x 10 + Bil[I]
      Pan jawab = Pan jawab + 1
      jawab[Pan jawab] = Sisa \ 13
```

Sebuah catatan kecil, untuk mencetak jawabannya, masih diperlukan ketelitian karena dalam ada kasus di mana jawabannya = 0 dan kasus di mana jawab[1] = 0 dan jawab[2] = 0.

Contoh 6.

Warisan (*Sumber : CEOI 1995*)

Dua orang bersaudara, Pak Dingklik dan Pak Dengklek ingin membagi beberapa barang warisan yang didapat dari almarhum orang tuanya. Setiap barang yang ada, harus diberikan kepada Pak Dingklik dan Pak Dengklek. Hanya saja, masalahnya setiap barang warisan tersebut tidak dapat dibagi menjadi dua. Supaya adil, masing-masing warisan tersebut diberikan sebuah nilai dalam bentuk bilangan bulat yang positif. Misalkan A dan B melambangkan total nilai dari keseluruhan warisan yang diterima oleh Pak Dingklik dan Pak Dengklek. Tujuan yang hendak dicapai adalah meminimalkan selisih dan A dan B. Buat sebuah program untuk menghitung nilai A dan B.

FORMAT MASUKAN

Baris pertama dari file WARIS.IN terdiri dari N, jumlah dari warisan yang ada (1<=N<=100). Baris berikutnya berisi N bilangan bulat positif, yang merupakan nilai dari setiap warisan. Tiap nilai <=200.

FORMAT KELUARAN

Tulis pada file WARIS.OUT nilai A dan B (A<B)

CONTOH MASUKAN

```
7
28 7 11 8 9 7 27
```

CONTOH KELUARAN

```
48 49
```

PEMBAHASAN
(Oleh Windra Swastika, Mantan peserta seleksi TOKI)

Jika masalah ini diselesaikan secara brute force (dengan mencoba semua kemungkinan yang ada), maka akan mempunyai kompleksitas sebesar O(2^N). Jika N=100, maka komputasi akan sangat memakan waktu yang besar. Untuk itu penyelesaian secara brute force adalah tidak disarankan untuk soal ini.

Pendekatan yang dilakukan adalah secara pemrograman dinamis (Dynamic Programming / DP). Konsep DP adalah menyelesaikan masalah dengan membagi menjadi sub masalah. Dari sub masalah yang terkecil (yang lebih mudah dicari nilai optimalnya), proses akan berlanjut untuk memecahkan masalah di atasnya (dengan memanfaatkan nilai optimal dari masalah sebelumnya). Nilai-nilai optimal tadi disimpan dalam sebuah tabel. DP menukar space dengan time!

Untuk soal ini, sebuah tabel yang mempunyai kombinasi penjumlahan untuk j=1, j=2, j=3 … j=N.
TABEL[j] = 0 jika j=0
TABEL[j] = 1..N artinya warisan yang terakhir yang diberikan untuk mendapatkan subtotal j
TABEL[j] = 101 artinya subtotal j tidak bisa dihasilkan

Misalkan W adalah array untuk menyimpan nilai dari setiap warisan. Nilai dari subtotal j akan ditambahkan untuk mendapatkan subtotal baru (j+W[I]) dengan syarat: TABEL[j]<i (nilai j adalah hasil dari subtotal yang pernah dicapai sebelumnya).

SOLUSI

```
{
  ===============
  -- WARIS.PAS --
  ===============
  Windra Swastika
}

const
  finput='waris.in';
  foutput='waris.out';
  kosong=101;

var
  tabel:array [0..10000] of byte;
  warisan:array [1..100] of byte;
  f:text;
  i:byte;
```

```
  j:word;
  n:byte;
  sum:word;

begin
  assign(f,finput);
  reset(f);
  readLn(f,n);
  for i:=1 to n do
  begin
    read(f,warisan[i]);
    inc(sum,g[i]);
  end;
  close(f);

  fillchar(tabel,sizeof(tabel),kosong);
  tabel[0]:=0;  {nilai awal untuk tabel[0]}
  for i:=1 to n do
    for j:=0 to sum shr 1-warisan[i] do
      if (tabel[j]<i) and (tabel[j+warisan[i]]=kosong) then
        tabel[j+warisan[i]]:=i;

  assign(f,foutput);
  rewrite(f);
  for i:=sum shr 1 downto 0 do  {cukup mulai dari setengah sum}
    if tabel[i]<>kosong then
      begin
        writeLn(i,' ',sum-i);
        break;
      end;

  close(f);
end.
```

Contoh 7.

Firetruck (Sumber : *ACM 1991 - Problem A*)

Departemen pemadam kebakaran kota Arjosari bekerja sama dengan dinas transportasi lokal untuk mengolah peta yang menunjukkan status jalan-jalan pada kota Arjosari. Sialnya, jalan-jalan di kota itu pada hari tertentu harus ditutup karena adanya pawai mingguan. Akibatnya, para petugas pemadam kebakaran harus mencari rute jalan yang paling pendek jika hendak memadamkan api pada jalan tertentu di kota Arjosari.

Jika pada suatu saat, departemen pemadam kebakaran menerima laporan terjadinya kebakaran pada jalan tertentu, maka pihak departemen pemadam kebakaran segera meminta daftar jalan-jalan yang dapat dilalui ke dinas transportasi. Dinas transportasi mengirimkan semua jalan yang pada saat itu bisa dilalui. Buat sebuah program untuk mencari semua rute yang diawali dari kantor pusat departemen pemadam kebakaran menuju jalan yang dituju.

FORMAT MASUKAN (fire.in)

Baris pertama berisi bilangan bulat yang merupakan jalan terdekat dari pusat api yang harus dituju (2<=N<21).

Baris-baris berikutnya merupakan pasangan bilangan bulat kurang dari 21 (dipisahkan dengan spasi) yang menunjukkan jalan yang dapat dilalui. Baris terakhir pada file input adalah pasangan bilangan 0 0. Misalkan pasangan jalan 4 7 ada pada file input, artinya bahwa jalan 4 dan 7 dapat dilalui, begitu juga sebaliknya.

FORMAT KELUARAN (fire.out)

Semua kemungkinan rute yang dapat menuju ke N yang dimulai dari 1.

Setiap baris berisi urutan jalan yang harus ditempuh diawali dari 1 dan diakhiri di N. Agar perjalanan petugas pemadam kebakaran efisien, truk pemadam kebakaran tidak boleh menempuh jalan yang pernah dilalui lebih dari sekali.

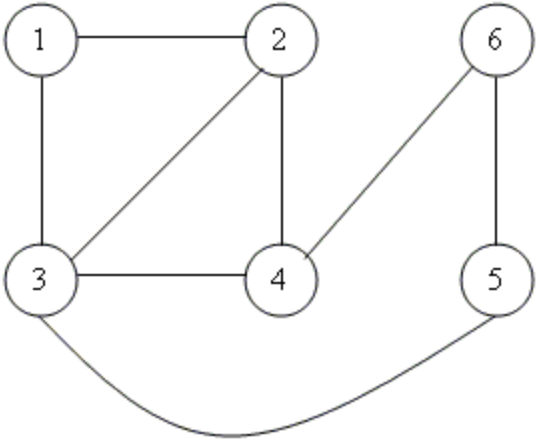
CONTOH MASUKAN

```
6
12
13
34
35
46
56
23
24
00
```

CONTOH KELUARAN

```
12346
12356
124356
1246
13246
1346
1356
```

Soal ini termasuk soal-soal yang menggunakan dasar teori graf. Pada teori graf istilah dikenal verteks dan edge. Yang dimaksud dengan verteks adalah titik dan edge adalah garis. Sebuah graf dinotasikan dengan e=[u,v], yang berarti bahwa edge e berawal pada verteks u dan berakhir pada verteks v. Pada Firetruck, terdapat maksimal 21 verteks. Sehingga contoh pada file input dapat dinotasikan e1=[1,2], yang artinya edge e1 berawal pada verteks 1 dan berakhir pada verteks 2. Dari file input, kita bisa mendapatkan sebuah graf seperti pada gambar berikut:



Dengan diagram seperti di atas, kita dapat membuat sebuah struktur data yang berisi jalur untuk masing-masing verteks. Dalam teori graf, jalur tersebut disebut dengan matriks jalur (*path matrix*).

```
const
  max_route=100
type
  tarrstreet=array[1..max_route,1..max_route] of boolean
var
  arrstreet:tarrstreet;
```

Variabel arrstreet adalah sebuah matriks jalur (array dengan tipe boolean) yang menunjukkan adanya jalur dari verteks ke verteks. Dengan struktur data tersebut serta menggunakan teknik pencarian DFS (Depth First Search), maka dapat dengan mudah dibuat prosedur penelusuran dari verteks awal (1) menuju verteks tertentu.

```
const
  finput='fire.in';
  foutput='fire.out';
  max_route=100;
{
  ===============
  -- FIRE.PAS --
  ===============
  Windra Swastika
}
```

```
type
  tstreet=1..21;
  troute=array[0..max_route] of tstreet;
  tarrstreet=array[1..max_route,1..max_route] of boolean;
var
  fi,fo:text;
  map:tarrstreet;
  used:array[1..max_route] of boolean;
  destination:tstreet;
  i,j:byte;
  N:byte;
  route:troute;

procedure seekroute(depth:byte;start:tstreet);
var
  i:byte;
begin
  if start=destination then
  begin
    for i:=0 to depth-1 do
      write(fo,route[i],' ');
    writeln(fo);
  end else
  for i:=1 to N do
  begin
    if map[start,i] and not used[i] then
    begin
      used[start]:=true;
      route[depth]:=i;
      seekroute(depth+1,i);
      used[start]:=false;
    end;
  end;
end;

begin
  assign(fi,'fire.in');
  reset(fi);
  readln(fi,destination);

  fillchar(map,sizeof(tarrstreet),false);
  fillchar(used,sizeof(used),false);
  fillchar(route,sizeof(troute),0);
  N:=0;  {number of street}

  repeat
    readln(fi,i,j);
    if (i>=0) and (j>=0) then
    begin
      map[i,j]:=true;
      map[j,i]:=true;
      inc(N);
    end;
```

```
  until (i=0) and (j=0);
  close(fi);

  assign(fo,'fire.out');
  rewrite(fo);
  route[0]:=1;    {start from street #1}
  used[1]:=true;  {street #1 already used}
  seekroute(1,1);

  close(fo);
end.
```