

Modul Praktikum

PEMROGRAMAN C++

LANJUTAN



Disusun oleh Komisi Teknik Komputer

Akademi Manajemen Informatika dan Komputer

BINA SARANA INFORMATIKA

2009

Kata Pengantar

Puji dan syukur penulis panjatkan kepada Allah SWT karena atas berkat dan rahmat-Nya Modul Praktikum Pemrograman C++ Lanjutan ini dapat kami susun. Modul ini merupakan kelanjutan materi praktikum pemrograman C++ yang telah dipelajari pada semester terdahulu pada Program Studi Diploma Tiga AMIK BSI jurusan Teknik Komputer.

Modul ini ditujukan khusus untuk mahasiswa jurusan Teknik Komputer, dengan pembahasan materi yang berhubungan dengan teknik-teknik pengaksesan perangkat keras yang telah tersedia pada komputer secara umum. Dengan memanfaatkan bahasa pemrograman C++ dan kompilator Borland C++ 5.02, mahasiswa diharapkan mampu membuat program komputer yang dapat memanipulasi perangkat keras komputer. Bahasa pemrograman C++ merupakan bahasa pemrograman yang berorientasi objek, oleh karena itu pembahasan pada modul ini tidak hanya menggunakan fungsi-fungsi pustaka yang dimiliki oleh Borland C++ 5.02 melainkan juga menggunakan struktur bahasa pemrograman berorientasi objek itu sendiri, seperti *class*, *constructor*, *destructor*, polimorfisme dan lain-lain. Diharapkan pembaca telah memahami konsep-konsep pemrograman berorientasi objek melalui praktikum pemrograman C++ pada semester terdahulu.

Pembahasan tentang teknik-teknik pengaksesan perangkat keras komputer akan menggunakan beberapa cara, diantaranya adalah dengan menggunakan fungsi-fungsi pustaka yang tersedia pada Borland C++ 5.02 dan membuat fungsi-fungsi tersendiri menggunakan teknik *inline assembly* memanfaatkan referensi arsitektur komputer Intel x86. Dengan memandang perangkat keras komputer sebagai suatu objek, maka pemrograman akan diutamakan pada pembuatan *class* yang memiliki atribut dan metode suatu objek. Contohnya adalah objek layar monitor memiliki atribut banyak baris dan kolom, warna karakter, jenis tampilan dan memiliki metode mencetak karakter, mengubah posisi kursor, mengubah jenis tampilan dan lain sebagainya.

Guna penyempurnaan modul ini pada masa yang akan datang, penulis berharap pembaca dan semua pihak yang menggunakan modul ini dapat memberikan saran-saran dan kritik yang sifatnya konstruktif. Akhir kata, penulis mengucapkan terima kasih kepada Bapak Ir. Naba Aji Notoseputro selaku Direktur Bina Sarana Informatika, Bapak Anton, S.Kom selaku Ketua Jurusan Teknik Komputer dan rekan-rekan Komisi Teknik Komputer serta semua pihak yang telah membantu penyusunan Modul Praktikum Pemrograman C++ Lanjutan ini.

Jakarta, Mei 2009

Penulis

Daftar Isi

Lembar Judul	i
Kata Pengantar	ii
Daftar Isi	iii

BAB I Dasar-Dasar Pengaksesan Perangkat Keras Komputer

Menggunakan Borland C++ 5.02

1.1. Pengenalan Register dan Interupsi pada Mikroprosesor	1
1.2. Pembuatan Project dan Kode Program	4
1.3. Memanggil Interupsi BIOS dan DOS Menggunakan Fungsi int86	7
1.4. Memanggil Interupsi BIOS dan DOS Menggunakan Teknik Inline Assembly	9
1.5. Latihan-latihan BAB I	10

BAB II Operasi Layar Modus Teks

2.1. Interupsi BIOS untuk Operasi Layar pada Modus Teks	11
2.2. Memilih Mode Video	11
2.3. Menampilkan Karakter dan Memindahkan Posisi Kursor	14
2.4. Membaca Karakter pada Posisi Kursor	18
2.5. ASCII Extended Character Set	19
2.5. Membuat Class untuk Operasi Layar pada Modus Teks	23
2.6. Latihan-latihan BAB II	xx

Bab III Input Menggunakan Keyboard

3.1. Interupsi BIOS untuk Input Melalui Keyboard
--

BAB I

Dasar-Dasar Pengaksesan

Perangkat Keras Komputer Menggunakan

Borland C++ 5.02

1.1. Pengenalan Register dan Interupsi pada Mikroprosesor

Setiap komputer memiliki sebuah bagian penting yang disebut *Central Processing Unit* (CPU) atau yang lebih dikenal dengan mikroprosesor. Sebagaimana halnya *Integrated Circuit* lainnya, mikroprosesor terdiri dari beberapa bagian kecil yang disebut dengan register. Pada arsitektur Intel x86 yang diperkenalkan oleh Intel Corporation pada tahun 1985, setiap mikroprosesor Intel memiliki beberapa jenis register berikut ini:

- a. General Purpose Register
- b. Pointer dan Index Register
- c. Segment Register
- d. Flag Register

Setiap jenis register memiliki fungsi-fungsi tersendiri. Berikut ini adalah rincian dari masing-masing register.

1. General Purpose Register

General Purpose Register terdiri dari register AX, BX, CX dan DX. Masing-masing register memiliki ukuran 16 bit (2 byte) dan masih dapat dibagi menjadi dua bagian, yaitu *high* dan *low*. Artinya register AX terdiri AH dan AL, demikian juga BX (BH dan BL), CX (CH dan CL) dan DX (DH dan DL). Masing-masing bagian *high* dan *low* berukuran 8 bit (1 byte). Register AX disebut juga akumulator dan berhubungan dengan operasi khusus seperti aritmatika, IN, OUT, *shift*, logika dan operasi *binary decimal*.

Register BX disebut juga *base register* dan berfungsi pada operasi *rotate*, aritmatika, *shift* dan logika. Register BX juga dapat digunakan untuk mereferensikan alamat memori. Register yang khusus digunakan untuk operasi perulangan (*loop*) dan pencacahan (*counter*) adalah register CX. Data register atau register DX digunakan pada operasi perkalian (MUL) dan menyimpan sebagian hasil perkalian 32 bit atau menyimpan nilai sisa dari operasi pembagian (DIV). Register DX juga digunakan pada operasi input/output suatu *port*.

Dalam mempelajari teknik-teknik pengaksesan perangkat keras menggunakan Borland C++ 5.02, General Purpose Register akan banyak digunakan. Oleh karena itu, sangatlah penting untuk memahami fungsi-fungsinya dari register ini.

2. Pointer dan Index Register

Pointer dan index register digunakan untuk menunjukkan suatu alamat memori, terdiri dari SP, BP, SI, DI dan IP. Register SP (*Stack Pointer*) dan BP (*Base Pointer*) berfungsi menunjukkan alamat *stack* saat terjadi operasi PUSH (menyimpan nilai ke dalam *stack*) dan POP (membaca nilai di dalam *stack*).

Register SI (*Source Index*) dan DI (*Destination Index*) digunakan pada saat operasi string jika kita membuat program dalam bahasa *assembly* murni. SI dan DI menyimpan nilai *offset* suatu string dalam segmen data memori. Register IP (*Instruction Pointer*) berfungsi menunjukkan alamat suatu instruksi program dalam memori saat program dijalankan. Register IP berpasangan dengan register CS (*Code Segment*) yang menyimpan semua kode program

dalam bentuk binari saat program dijalankan dan dimuat dalam memori. Register IP dan CS dituliskan dalam notasi CS:IP, misalkan:

```
0F6C:0100 mov ah, 02  
0F6C:0102 mov bl, 05
```

Arti instruksi diatas adalah saat program dijalankan register CS menunjukan alamat 0F6C heksadesimal dan register IP menunjukan alamat 0100 heksadesimal. Instruksi yang tersimpan adalah mov ah, 02 (simpan nilai 2 heksadesimal ke register AH). Hal yang sama juga terjadi pada baris kedua, pada baris kedua nilai register IP berubah menjadi 0102 yang sebelumnya 0100. Hal ini terjadi karena instruksi mov pada baris pertama menggunakan memori sebesar 2 byte.

Notasi CS:IP dapat dianalogikan dengan tabel atau matriks, dimana CS menunjukan nomor baris sedangan register IP menunjukan nomor kolom. Dalam membuat program menggunakan bahasa pemrograman C++ seorang programmer tidak harus mengubah nilai-nilai CS:IP karena nilai-nilai dalam kedua register tersebut diatur oleh kompilator. Merupakan hal yang beresiko mengubah nilai-nilai pada register CS:IP karena pasangan register tersebut menunjukan alamat instruksi. Kesalahan mengubah nilai register CS:IP akan menyebabkan program berhenti melakukan eksekusi atau program mengalami *hang*.

3. Segment Register

Segment register terdiri dari CS (Code Segment), DS (Data Segment), SS (Stack Segment) dan ES (Extra Segment) yang masing-masing berukuran 16 bit (2 byte). Register CS berpasangan dengan IP berfungsi menyimpan alamat instruksi, register DS berpasangan dengan register DX (DS:DX) yang menyimpan alamat data. Register SS (Stack Segment) menyimpan alamat memori stack sedangkan ES (Extra Segment) menyimpan alamat segment tambahan.

Dalam pemrograman C++ segment register umumnya tidak perlu diakses secara langsung karena segment register telah diatur oleh kompilator.

4. Flag Register

Pada arsitektur awal Intel x86 terdapat beberapa flag register, yaitu:

- ❑ CF (Carry Flag), menandakan jika suatu instruksi ADD menghasilkan nilai *carry*, atau suatu instruksi SUB menghasilkan nilai *borrow*
- ❑ PF (Parity Flag), menandakan jika suatu instruksi menghasilkan nilai genap atau ganjil. Register ini akan bernilai 1 jika bilangan yang dihasilkan bernilai genap.
- ❑ AF (Auxiliary Carry Flag), digunakan untuk operasi desimal berkode binari (BCD), seperti pada operasi AAA (ASCII Adjust for Addition).
- ❑ OF (Overflow Flag), menandakan jika suatu operasi aritmatika mengalami overflow (melebihi jangkauan nilai yang telah ditentukan atau hasil operasi aritmatika melebihi ukuran register).
- ❑ SF (Sign Flag), digunakan pada operasi aritmatika yang menggunakan bilangan bertanda (bilangan positif atau bilangan negatif).
- ❑ ZF (Zero Flag), menandakan jika suatu operasi aritmatika menghasilkan nilai nol.
- ❑ DF (Direction Flag), digunakan pada operasi string yang menunjukan arah proses.
- ❑ IF (Interrupt Enable Flag), menandakan jika CPU akan memproses interupsi jika bernilai 1 atau mengabaikan interupsi jika bernilai nol.
- ❑ TF (Trap Flag), digunakan saat *debugging*, dengan mode *single step*.
- ❑ NT (Nested Task), digunakan untuk memantau jalannya interupsi yang terjadi secara beruntun.
- ❑ IOPL (I/O Protection Level), flag ini digunakan jika program berjalan pada mode

- terproteksi (*protected mode*).
- ❑ PE (Protection Enable), flag ini digunakan untuk mengaktifkan mode terproteksi.
 - ❑ MP (Monitor Coprocessor), digunakan untuk memantau kerja *coprocessor* dan menangani terjadinya instruksi WAIT.
 - ❑ EM (Emulate Coprocessor), digunakan jika prosesor akan mengemulasikan kerja *coprocessor*.
 - ❑ ET (Extention Type), digunakan untuk menentukan jenis *coprocessor* (80287 atau 80387).
 - ❑ VF (Virtual 8086 Mode), digunakan jika ingin menjalankan aplikasi *real mode* pada *protected mode*.

Pada pemrograman C++, nilai-nilai pada Flag Register tidak diberikan secara manual oleh programmer, tetapi diatur oleh sistem operasi pada saat program C++ dijalankan.

Register-register yang telah dibahas sebelumnya hanya menyimpan data dan jenis instruksi yang akan dijalankan oleh mikroprosesor. Lalu pertanyaannya adalah bagaimana memerintahkan mikroprosesor melaksanakan instruksi tertentu. Jawabnya adalah dengan melakukan interupsi (*interrupt*). Interupsi adalah permintaan kepada mikroprosesor untuk melakukan tugas tertentu, ketika terjadi interupsi maka mikroprosesor menghentikan dahulu instruksi yang sedang dikerjakan dan menjalankan tugas yang diminta sesuai dengan interupsi yang diberikan.

Terdapat dua jenis interupsi, yaitu interupsi BIOS dan interupsi DOS. Interupsi BIOS (Basic Input Output System) ditanam pada chip ROM BIOS oleh pabrik komputer sedangkan interupsi DOS (Disk Operating System) hanya dapat digunakan jika komputer menjalankan sistem operasi DOS atau Microsoft Windows. Berikut ini adalah tabel yang menjelaskan daftar interupsi BIOS dan DOS.

Tabel I.1. Daftar Interupsi BIOS

Nomor Interupsi	Nama Interupsi	Nomor Interupsi	Nama Interupsi
00h	Divide By Zero	10h	Video Service
01h	Single Step	11h	Equipment Check
02h	Non-Maskable Interrupt (NMI)	12h	Memory Size
03h	Break Point	13h	Disk Service
04h	Arithmatic Overflow	14h	Communication (RS-232)
05h	Print Screen	15h	Cassette Service
06h	Reserved	16h	Keyboard Service
07h	Reserved	17h	Printer Service
08h	Clock Tick (Timer)	18h	ROM BASIC
09h	Keyboard	19h	Bootstrap Loader
0ah	I/O Channel Action	1ah	BIOS Time and Date
0bh	COM 1 (serial 1)	1bh	Control Break
0ch	COM 2 (serial 2)	1ch	Timer Tick
0dh	Fixed Disk	1dh	Video Initialization
0eh	Diskette	1eh	Disk Parameters
0fh	LPT 1 (Parallel 1)	1fh	Graphics Character

Tabel I.2. Daftar Interupsi DOS

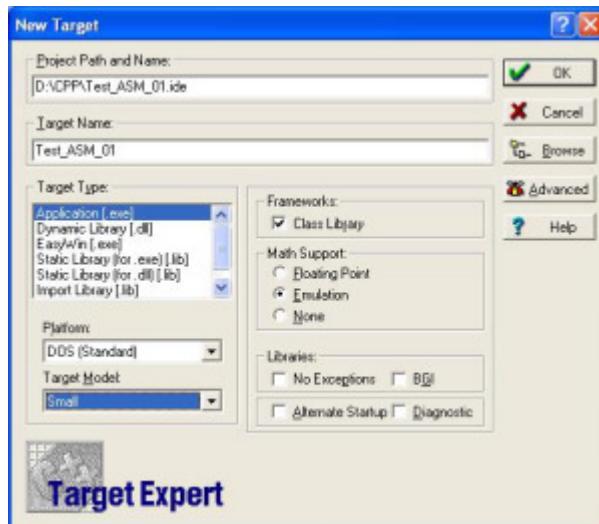
Nomor Interupsi	Nama Interupsi
20h	Terminate Program
21h	DOS Function Services
22h	Terminate Code
23h	Ctrl-Break Code
24h	Critical Error Handler
25h	Absolute Disk Read
26h	Absolute Disk Write
27h	Terminate But Stay Resident

Pada kedua tabel diatas, setelah nomor interupsi selalu diakhiri oleh huruf h. Maksud dari huruf tersebut adalah nomor-nomor interupsi yang digunakan diberikan dalam bentuk bilangan heksadesimal. Pada saat membuat program yang sesungguhnya menggunakan Borland C++ 5.02 programmer dapat mengetikkan nomor interupsi dalam bentuk bilangan desimal (basis 10), oktal (basis 8) atau heksadesimal (basis 16). Penulisan ketiga bentuk bilangan tersebut mengikuti aturan yang ditetapkan oleh *grammar* bahasa C++, yaitu dituliskan sebagaimana biasanya jika menggunakan bilangan desimal (misalnya: 10, 45, dan 150), diawali dengan karakter “0” jika menggunakan bilangan oktal (misalnya: 012, 055, dan 0226) dan diawali dengan karakter “0x” jika menggunakan bilangan heksadesimal (misalnya 0x0a, 0x2d, dan 0x96).

1.2. Pembuatan Project dan Kode Program

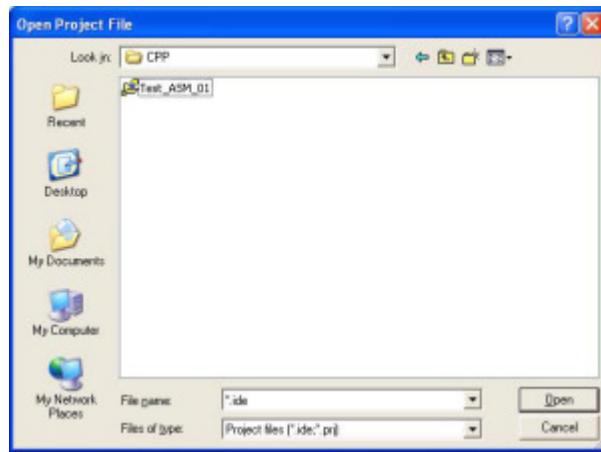
Untuk bisa membuat program menggunakan fungsi `int86(...)` dan teknik *inline assembly*, maka kode program C++ yang akan dibuat harus merupakan bagian dari sebuah *project file*, tidak bisa hanya sebuah file kode sumber saja.

1. Klik menu **File** pada menu bar, kemudian pilih dan klik menu **New** dan pilih menu **Project**, maka akan muncul jendela New Target seperti pada gambar I.1 berikut ini.



Gambar I.1. Jendela New Target

2. Klik tombol **Browse** untuk menentukan nama file dan lokasi penyimpanan project seperti diperlihatkan pada gambar I.2 berikut.



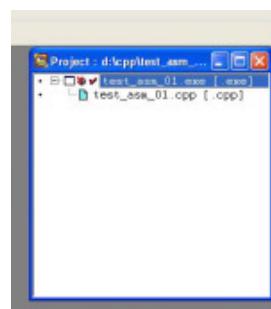
Gambar I.2. Jendela Open Project File

3. Ketikan nama project pada *textbox* File name dan klik tombol **Open**. Textbox Project Path and Name pada jendela New Target akan berisi nama direktori dimana project disimpan dan Target Name akan berisi nama project.
4. Pada *listbox* Target Type pilihlah **Application (.exe)**.
5. Pada *combo box* Platform pilihlah **DOS (Standard)**.
6. Pada *combo box* Target Model pilihlah **Small**.
7. Berilah tanda cek pada *check box* **Class Library**.
8. Pada *radio button* Math Support pilihlah sesuai dengan kondisi berikut ini:

❑ Jika program yang akan dibuat memerlukan perhitungan *floating point* (bilangan pecahan) bisa memilih **Emulation** (operasi aritmatika *floating point* dilakukan menggunakan emulasi perangkat lunak dengan konsekuensi program menjadi lebih lambat) atau pilihlah **Floating Point** (operasi aritmatika dilakukan dengan memanfaatkan *arithmetic co-processor* pada mikroprosesor, eksekusi program menjadi lebih cepat).

❑ Jika program yang dibuat tidak membutuhkan perhitungan *floating point* maka pilihlah **None**.

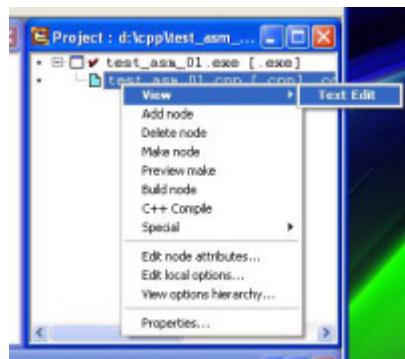
9. Kosongkan semua *check box* pada pilihan Libraries.
10. Klik tombol **OK** untuk mulai membuat program. Borland C++ 5.02 akan menampilkan Project Window seperti terlihat pada gambar I.3 berikut ini.



Gambar I.3. Project Window Borland C++ 5.02

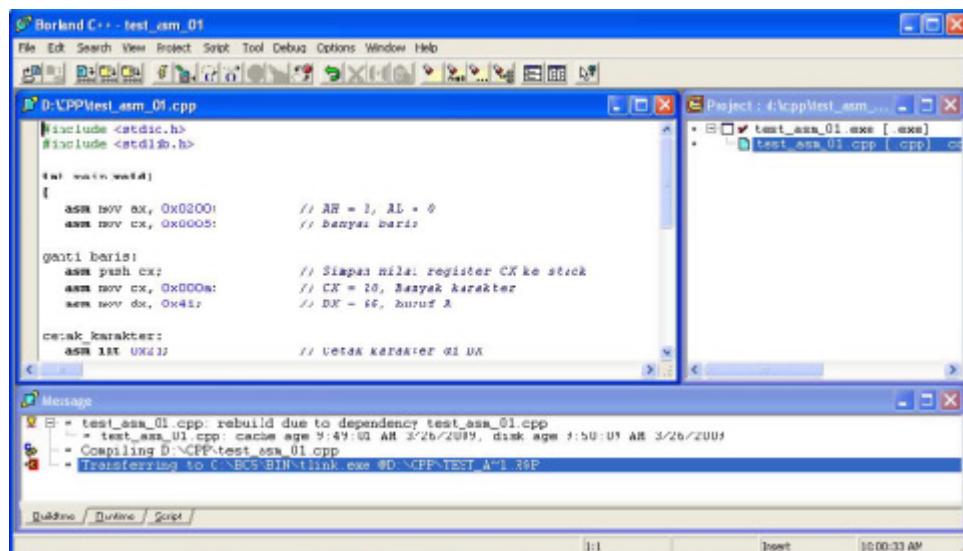
11. Pada Project Window terdapat node nama project (berakhiran **.exe**) dan node kode

program (berakhiran .cpp) dibawah node nama project. Klik kanan pada node kode program sampai muncul menu *pop-up* dan pilihlah menu **View** kemudian pilih sub menu **Text Edit** seperti pada gambar I.4 berikut ini.



Gambar I.4. Node kode program pada Project Window

12. Borland C++ 5.02 akan memunculkan Edit Window seperti diperlihatkan pada gambar I.5. Pada Edit Window inilah kode program mulai diketikan.



Gambar I.5. Edit Window pada Borland C++ 5.02

13. Untuk menutup project dapat dilakukan dengan mengklik icon [X] pada Project Window dan Edit Window.
14. Untuk membuka kembali file project yang sudah ada dapat dilakukan dengan memilih menu **Project** dari menu bar dan memilih menu **Open Project**, maka akan ditampilkan jendela Open Project File seperti gambar I.6 berikut ini.



Gambar I.6. Jendela Open Project File

15. Untuk membuka project, kliklah nama file project dari *listbox* File Name dan klik tombol **OK**.

1.3. Memanggil Interupsi BIOS atau DOS Menggunakan Fungsi int86

Borland C++ 5.02 menyediakan sebuah fungsi untuk menjalankan interupsi BIOS atau DOS, yaitu fungsi int86. Nama fungsi ini dideklarasikan di file header dos.h, berikut ini adalah sintaks dari fungsi int86:

```
int int86(int nomor, union REGS *inregs,
           union REGS *outregs)
```

Keterangan:

- ❑ **int nomor**, nomor interupsi yang akan dijalankan.
- ❑ **union REGS *inregs**, representasi register pada mikroprosesor yang berisi nilai yang akan digunakan untuk menjalankan interupsi.
- ❑ **union REGS *outregs**, representasi register pada mikroprosesor setelah interupsi dijalankan.

Pada fungsi int86, parameter nomor bertipe integer dan dikirimkan secara nilai (*by value*), sedangkan parameter inregs dan outregs merupakan tipe data union REGS yang telah didefinisikan pada header dos.h dan dikirimkan secara acuan (*by reference*). Diharapkan pembaca telah memahami tipe data union dan struktur serta pengiriman parameter secara acuan atau secara nilai. Fungsi int86 mengembalikan nilai bertipe integer dari register AX setelah interupsi dijalankan. Tipe data union REGS yang didefinisikan pada file header dos.h merupakan union yang terdiri dari struktur WORDREGS dan BYTEREGS. Berikut ini adalah deklarasi union REGS pada file header dos.h :

```
union REGS {
    struct WORDREGS  x;
    struct BYTEREGS h;
};
```

Sedangkan deklarasi struktur WORDREGS dan BYTEREGS adalah sebagai berikut:

```
struct BYTEREGS {
    unsigned char al, ah, bl, bh;
    unsigned char cl, ch, dl, dh;
};

struct WORDREGS {
    unsigned int ax, bx, cx, dx;
    unsigned int si, di, cflag, flags;
};
```

Dari perincian diatas, dapat disimpulkan bahwa struktur BYTEREGS merepresentasikan register-register pada General Purpose Register yang berukuran 8 bit (1 byte), yaitu AL, AH, BL, BH, CL, CH, DL dan DH. Sedangkan struktur WORDREGS merepresentasikan register-register pada General Purpose Register yang berukuran 16 bit (2 byte) ditambah register SI, DI dan Flag Register. Struktur WORDREGS maupun BYTEREGS menempati lokasi memori yang sama dalam union REGS.

Berikut ini adalah contoh sederhana penggunaan int86 untuk mencetak karakter menggunakan interupsi BIOS 10 heksadesimal servis 09 heksadesimal. Simpan contoh dibawah ini dengan nama project contoh01.ide dan kode program contoh01.cpp.

contoh01.cpp:

```
01 #include <conio.h>
02 #include <dos.h>
03 #include <stdlib.h>
04
05 #define VIDEO_INT 0x10 /* BIOS Video Interrupt      */
06
07 int main(void)
08 {
09     union REGS in, out; /* Deklarasi variabel      */
10
11     in.h.ah = 0x09;    /* AH = 9 heksadesimal      */
12     in.h.al = 'A';    /* AL = 41 heksadesimal, huruf A */
13     in.h.bh = 0x00;    /* BH = 0, halaman video      */
14     in.h.bl = 0x07;    /* BL = 7, warna huruf dan dasar */
15     in.h.ch = 0x00;    /* CH dan CL menentukan banyak */
16     in.h.cl = 0x01;    /* huruf yang akan dicetak      */
17
18     clrscr();
19     int86(VIDEO_INT, &in, &out);
20     getch();
21
22     return EXIT_SUCCESS;
23 }
```

Baris 1 sampai baris 3 pada program diatas merupakan penentuan header yang akan digunakan, yaitu conio.h, dos.h dan stdlib.h. Header conio.h digunakan karena pada baris 18 kita akan menggunakan fungsi clrscr dan pada baris 20 kita akan menggunakan fungsi getch. Header dos.h digunakan karena pada baris 9 kita mendeklarasikan variabel in dan out menggunakan tipe union REGS serta pada baris 19 kita akan memanggil fungsi int86. Header stdlib.h digunakan karena pada baris 22 kita akan menggunakan nilai konstanta EXIT_SUCCESS.

Pada baris 5 kita mendeklarasikan konstanta dengan nama VIDEO_INT yang bernilai 10 heksadesimal. Konstanta ini digunakan saat memanggil fungsi video int86 untuk menjalankan interupsi 10 heksadesimal servis 9 heksadesimal. Cobalah untuk mengganti nilai pada in.h.bl dari yang semula 0x07 menjadi nilai lain, misalnya 0x0a, 0x09 atau 0x04 kemudian jalankan program. Apa yang terjadi?

Pada contoh diatas variabel yang digunakan bernama in dan out. Perhatikanlah cara memberikan nilai pada variabel in yang merupakan union REGS pada baris 11 sampai baris 16. Pada contoh diatas in.h.ah, atau in.h.al dan sebagainya berarti memberikan nilai union REGS pada struktur BYTEREGS dengan nama h dan field ah atau al yang bertipe unsigned character.

1.4. Memanggil Interupsi BIOS dan DOS Menggunakan Teknik Inline Assembly

Teknik *inline assembly* adalah teknik menuliskan kode-kode bahasa *assembly* diantara kode-kode bahasa pemrograman yang lain seperti Pascal dan C/C++. Tidak semua kompilator memiliki kemampuan mengkompilasi kode inline assembly, Borland C++ dan Turbo C/C++ memiliki kemampuan ini. Cara menggunakan teknik inline assembly pada Borland C++ 5.02 adalah dengan menggunakan kata cadangan **asm** dan diikuti oleh kode program bahasa assembly serta diakhiri dengan tanda semicolon/titik koma (;). Berikut sintaks penggunaan kata kunci **asm**:

```
asm <opcode> <operand>; /* Komentar bahasa C/C++ */
```

Kata kunci **asm** bisa juga diketikkan sebagai blok pernyataan, seperti berikut ini:

```
asm {  
    <opcode> <operand>; /* Komentar bahasa C/C++ */  
    <opcode> <operand>; /* atau baris baru */  
}
```

Berikut ini adalah program dengan tujuan yang sama seperti pada contoh01.cpp untuk mencetak huruf A di layar. Perbedaannya adalah fungsi int86 digantikan dengan kode-kode inline assembly, berikut ini adalah kode programnya yang disimpan dengan nama file project contoh02.ide dan file kode program contoh02.cpp.

contoh02.cpp:

```
01 #include <conio.h>  
02 #include <dos.h>  
03 #include <stdlib.h>  
04  
05 #define VIDEO_INT 0x10 // BIOS Video Interrupt  
06  
07 int main(void)  
08 {  
09     clrscr();  
10  
11     asm mov ah, 0x09; // AH = 9 heks.  
12     asm mov al, 0x41; // AL = 41 heks., huruf A  
13     asm mov bh, 0x00; // BH = 0 heks., halaman layar  
14     asm mov bl, 0x07; // BL = 7 heks., warna huruf  
15     asm mov ch, 0x00; // CH dan CL menentukan banyak  
16     asm mov cl, 0x01; // huruf yang akan dicetak  
17     asm int VIDEO_INT; // Interupsi BIOS 10h  
18  
19     getch();  
20     return EXIT_SUCCESS;  
21 }
```

Perhatikanlah baris 11 sampai dengan baris 16! Instruksi `mov ah, 0x09` artinya menyimpan nilai 9 heksadesimal kedalam register AH. Demikian juga dengan instruksi `mov bh, 0x00` atau `mov cl, 0x01`. Dalam pemrograman bahasa assembly *opcode* `mov` digunakan untuk memberikan nilai kedalam suatu operand, dalam hal ini register AH, AL, BH, BL dan sebagainya. Kemudian amati pula baris 17. Pada baris 17 instruksi `int VIDEO_INT` bukan berarti mendeklarasikan variabel `VIDEO_INT` dengan tipe data integer, melainkan menjalankan interupsi yang nilai interupsinya disimpan dalam konstanta `VIDEO_INT`, yaitu

0x10 atau 10 heksadesimal. Dalam pemrograman bahasa assembly opcode **int** digunakan untuk menjalankan interupsi sesuai dengan nomor interupsi sesudah kata kunci int.

1.5. Latihan-latihan BAB I

1. Apa yang dimaksud dengan register?
2. Apa yang dimaksud dengan interupsi? Sebutkan jenis-jenisnya!
3. Ada berapa cara menjalankan interupsi pada Borland C++ 5.02? Jelaskan!
4. Berikut ini adalah data-data yang diketahui untuk mencetak huruf/karakter menggunakan interupsi BIOS 10h:

- Register AH berisi nilai 9 heksadesimal.
- Register AL berisi kode ASCII dari huruf/karakter yang akan dicetak.
- Register BH berisi halaman layar, halaman pertama nilainya 0.
- Register BL berisi nilai warna huruf/karakter (warna asal adalah 7).
- Register CH dan CL menentukan banyaknya karakter pada AL akan dicetak.
Jika cuma 1 huruf, maka CH = 0 dan CL = 1.

Buatlah sebuah program C++ yang dapat mencetak huruf Z dengan warna biru menggunakan fungsi int86 !

5. Selain menggunakan interupsi BIOS 10h, untuk mencetak karakter juga bisa menggunakan interupsi DOS 21h. Berikut ini adalah prosedurnya:

- Register AH harus bernilai 2 heksadesimal.
- Register DL berisi kode ASCII dari huruf/karakter yang akan dicetak.

Buatlah sebuah program C++ yang dapat mencetak huruf Q dilayar menggunakan teknik inline assembly!

BAB II

Operasi Layar Modus Teks

2.1. Interupsi BIOS untuk Operasi Layar pada Modus Teks

Untuk melakukan operasi-operasi pada layar seperti memilih mode video, menampilkan karakter dan lain-lain, BIOS telah menyediakan nomor interupsi khusus, yaitu interupsi 10 heksadesimal. Operasi-operasi yang akan dilakukan sebelum menjalankan interupsi ini ditentukan oleh nilai yang disimpan dalam register AH.

2.2. Memilih Mode Video

Mode video adalah cara layar monitor menampilkan output, apakah output yang ditampilkan dalam bentuk matriks-matriks teks atau dalam bentuk picture element (*pixel*). Pada sub bab ini akan dibahas cara menggunakan mode video teks. Mode teks sendiri memiliki beberapa mode lain yang dapat dipilih, normalnya mode yang digunakan pada layar monitor masa kini adalah mode teks 25 baris 80 kolom dan mampu menampilkan 16 warna yang berbeda. Tabel II.1 berikut menjelaskan beberapa mode video teks.

Tabel II.1. Mode video teks

Nomor (heksadesimal)	Mode
00	Hitam-putih, 25 baris, 40 kolom
01	16 warna, 25 baris, 40 kolom
02	16 warna gray text, 25 baris, 80 kolom
03	16 warna, 25 baris, 80 kolom
07	Monochrome, 25 baris, 80 kolom

Prosedur untuk memilih mode video menggunakan interupsi 10 heksadesimal adalah sebagai berikut:

- Register AH harus bernilai 0.
- Register AL berisi nomor dari mode video yang akan digunakan.

Sebagai contoh, berikut ini adalah program untuk memilih mode video 01 heksadesimal (16 warna, 25 baris dan 40 kolom) kemudian mengembalikannya menjadi mode video normal menggunakan fungsi int86. Simpan project berikut ini dengan nama contoh03.ide dan nama file kode program contoh03.cpp.

contoh03.cpp:

```
01 #include <conio.h>
02 #include <dos.h>
03 #include <stdio.h>
04 #include <stdlib.h>
05
06 #define VIDEO_INT 0x10          // Nomor interupsi 10h
07 #define UCHAR unsigned char
08
09 void setMode(UCHAR mode);    // Deklarasi fungsi untuk
10                           // mengubah mode video
11 int main(void)
12 {
```

```

13 printf("Tekan ENTER untuk mengubah mode...\n");
14 getch();
15
16 setMode(0x01);           // Ubah mode video
17 printf("Mode 01 heksadesimal.\n"); // Informasi
18 printf("Tekan ENTER kembali ke mode normal..."); 
19 getch();
20
21 setMode(0x03);           // Kembali ke mode normal
22 printf("Mode normal\n");
23 getch();
24
25 return EXIT_SUCCESS;
26 }
27
28 void setMode(UCHAR mode)
29 {
30     union REGS in, out;          // Deklarasi variabel
31
32     in.h.ah = 0x00;             // Register AH = 0
33     in.h.al = mode;            // Register AL = mode
34
35     int86(VIDEO_INT, &in, &out); // Jalankan interupsi
36     return;
37 }
```

Pada contoh diatas, setiap terjadi pergantian mode video akan selalu menimbulkan efek *clear screen*. Bagaimana cara menghilangkan efek clear screen ini? Jawabnya adalah dengan menset bit ke-7 pada register AL menjadi 1. Berikut ini adalah contoh yang sama seperti pada contoh03.cpp, namun bit ke-7 pada register AL akan diset menjadi 1 dan menggunakan teknik inline assembly. Simpan project berikut dengan nama contoh04.ide dan nama file kode program contoh04.cpp.

contoh04.cpp:

```

01 #include <conio.h>
02 #include <dos.h>
03 #include <stdio.h>
04 #include <stdlib.h>
05
06 #define VIDEO_INT 0x10          // Nomor interupsi 10h
07 #define UCHAR unsigned char
08
09 void setMode(UCHAR mode);    // Deklarasi fungsi untuk
10                           // mengubah mode video
11 int main(void)
12 {
13     printf("Tekan ENTER untuk mengubah mode...\n");
14     getch();
15
16     setMode(0x01);           // Ubah mode video
17     printf("Mode 01 heksadesimal.\n"); // Informasi
18     printf("Tekan ENTER kembali ke mode normal..."); 
19     getch();
20
21     setMode(0x03);           // Kembali ke mode normal
22     printf("Mode normal\n");
```

```

23     getch();
24
25     return EXIT_SUCCESS;
26 }
27
28 void setMode( UCHAR mode)
29 {
30     asm mov ah, 0x00;      // Register AH = 0
31     asm mov al, mode;      // Register AL = mode
32     asm or al, 0x80;       // OR-kan dengan 80 heksadesimal
33     asm int VIDEO_INT;    // Lakukan interupsi
34
35     return;
36 }
```

Kedua contoh program sebelumnya digunakan untuk mengubah mode video. Bagaimana jika kita tidak mengetahui mode video yang sedang digunakan? Jawabnya adalah dengan menjalankan interupsi 10 heksadesimal servis 0f heksadesimal. Setelah interupsi ini dijalankan register AH berisi banyaknya kolom, register AL berisi nomor mode video yang digunakan dan register BH berisi nomor halaman tampilan yang digunakan. Berikut ini adalah contoh programnya, simpan project berikut ini dengan nama file contoh05.ide dan kode program contoh05.cpp.

contoh05.cpp:

```

01 #include <conio.h>
02 #include <dos.h>
03 #include <stdio.h>
04 #include <stdlib.h>
05
06 #define VIDEO_INT 0x10
07
08 void getMode(union REGS *reg);
09
10 int main(void)
11 {
12     union REGS layar;
13
14     getMode(&layar);
15
16     printf("Informasi Layar Monitor\n");
17     printf("Banyak kolom\t\t: %d\n", layar.h.ah);
18     printf("Nomor mode\t\t: %0x\n", layar.h.al);
19     printf("Halaman tampilan\t: %d\n", layar.h.bh);
20     getch();
21
22     return EXIT_SUCCESS;
23 }
24
25 void getMode(union REGS *reg)
26 {
27     union REGS *in;
28
29     in->h.ah = 0x0f;
30
31     int86(VIDEO_INT, in, reg);
32 }
```

```
33     return;  
34 }
```

2.3. Menampilkan Karakter dan Memindahkan Posisi Kursor

Program contoh01.cpp dan contoh02.cpp pada bab 1 merupakan contoh program untuk menampilkan karakter/huruf pada layar monitor. Pada sub bab ini akan dibahas cara menampilkan karakter menggunakan interupsi 10 heksadesimal servis 09 heksadesimal secara lebih mendalam. Untuk lebih jelaskannya, berikut ini adalah prosedur untuk menjalankan interupsi 10 heksadesimal servis 09 heksadesimal:

- ❑ Register AH berisi nilai 9 heksadesimal.
- ❑ Register AL berisi kode ASCII dari huruf/karakter yang akan dicetak.
- ❑ Register BH berisi halaman layar, halaman pertama nilainya 0.
- ❑ Register BL berisi nilai warna huruf/karakter (warna asal adalah 7).
- ❑ Register CH dan CL menentukan banyaknya karakter pada AL akan dicetak.
Jika cuma 1 huruf, maka CH = 0 dan CL = 1.

Permasalahan mencetak karakter menggunakan interupsi 10 heksadesimal servis 09 heksadesimal adalah setelah karakter ditampilkan dilayar, posisi kursor tidak berpindah ke kolom berikutnya. Akibatnya adalah ketika karakter berikutnya akan ditampilkan maka karakter yang sebelumnya akan tertimpa dengan karakter yang baru. Solusi untuk mengatasi permasalahan ini adalah sebelum karakter ditampilkan kita harus mengetahui posisi kursor, kemudian mencetak karakter tersebut dan mengubah posisi kursor setelah karakter dicetak. Untuk melakukan hal tersebut maka kita harus tahu cara mengetahui posisi kursor dan cara memindahkan posisi kursor.

Untuk mengetahui posisi kursor dapat menggunakan interupsi 10 heksadesimal servis 03 heksadesimal. Berikut ini adalah prosedur untuk menjalakan interupsi 10 heksadesimal servis 03 heksadesimal:

- ❑ Register AH harus bernilai 3 heksadesimal.
- ❑ Register BH berisi nomor halaman tampilan, halaman pertama nilainya 0.

Setelah interupsi dilakukan maka register DH berisi nomor baris dan register DL berisi nomor kolom. Sedangkan untuk memindahkan posisi kursor adalah dengan menggunakan interupsi 10 heksadesimal servis 02 heksadesimal. Berikut ini adalah prosedurnya:

- ❑ Register AH harus bernilai 2 heksadesimal.
- ❑ Register BH berisi nomor halaman tampilan, halaman pertama nilainya 0.
- ❑ Register DH berisi nomor baris (dimulai dari 0 sampai 24).
- ❑ Register DL berisi nomor kolom (dimulai dari 0 sampai batas akhir dikurangi 1).

Berikut ini adalah contoh program untuk menampilkan huruf A dan Z dengan warna dasar biru dan warna huruf putih. Simpan project berikut ini dengan nama file contoh06.ide dan nama file kode program contoh06.cpp:

contoh06.cpp:

```
01 #include <conio.h>  
02 #include <dos.h>  
03 #include <stdlib.h>  
04  
05 #define VIDEO_INT 0x10  
06 #define UCHAR unsigned char
```

```

07
08 void getCursorPos(UCHAR *y, UCHAR *x);
09 void setCursorPos(UCHAR y, UCHAR x);
10 void writeChar(UCHAR letter, UCHAR attr);
11
12 int main(void)
13 {
14     UCHAR baris, kolom;
15
16     getCursorPos(&baris, &kolom); // Baca posisi kursur
17     writeChar('A', 0x1f); // Cetak huruf A
18     setCursorPos(baris, ++kolom); // Pindahkan kurSOR
19     writeChar('Z', 0x1f); // Cetak huruf Z
20     setCursorPos(baris, ++kolom); // Pindahkan kurSOR
21     getch();
22
23     return EXIT_SUCCESS;
24 }
25
26 void getCursorPos(UCHAR *y, UCHAR *x) // Baca posisi
27 { // kurSOR
28     UCHAR row, col;
29
30     asm mov ah, 0x03; // Register AH = 3 heksadesimal
31     asm mov bh, 0x00; // Register BH = 0 heksadesimal
32     asm int VIDEO_INT; // Lakukan interupsi
33     asm mov row, dh; // Salin register DH ke row
34     asm mov col, dl; // Salin register DL ke col
35
36     *y = row; *x = col; // Salin row ke y, col ke x
37
38     return;
39 }
40
41 void setCursorPos(UCHAR y, UCHAR x) // Memindahkan
41 { // Posisi kurSOR
42     asm mov ah, 0x02; // Register AH = 3 heksadesimal
43     asm mov bh, 0x00; // Register BH = 0 heksadesimal
44     asm mov dh, y; // Register DH = letak baris
45     asm mov dl, x; // Register DL = letak kolom
46     asm int VIDEO_INT; // Lakukan interupsi
47
48     return;
49 }
50
51 void writeChar(UCHAR letter, UCHAR attr) // Mencetak
52 { // huruf
53     asm mov ah, 0x09; // Register AH = 9 heksadesimal
54     asm mov al, letter; // Register AL = hurufnya
55     asm mov bh, 0x00; // Register BH = 0 heksadesimal
56     asm mov bl, attr; // Register BL = warna huruf
57     asm mov ch, 0x00; // Register CH dan CL menentukan
58     asm mov cl, 0x01; // banyak pencetakan
59     asm int VIDEO_INT; // Lakukan interupsi
60
61     return;
62 }

```

Program diatas terdiri dari empat fungsi, yaitu fungsi main, getCursorPos, setCursorPos dan writeChar. Fungsi getCursorPos berguna untuk mengetahui posisi kursor, fungsi ini mengirimkan parameter y dan x secara acuan. Setelah pemanggilan fungsi, parameter x menyimpan posisi kolom kursor sedangkan parameter y menyimpan posisi baris kursor. Fungsi getCursorPos digunakan untuk memindahkan posisi kursor, fungsi ini mengirimkan parameter y dan x secara nilai. Parameter y digunakan untuk menentukan posisi baris sedangkan parameter x untuk menentukan posisi kolom kursor. Fungsi getCursorPos hampir mirip dengan fungsi wherex dan wherey milik Borland C++ atau Turbo Pascal, sedangkan fungsi setCursorPos hampir mirip dengan fungsi gotoxy.

Fungsi writeChar digunakan untuk menampilkan karakter, fungsi ini mengirimkan parameter letter dan attr secara nilai. Parameter letter berisi karakter yang akan ditampilkan sedangkan parameter attr menentukan atribut karakter (warna karakter dan warna dasar). Fungsi ini tidak mengubah posisi kursor, oleh karena itu sesudah pemanggilan fungsi, untuk memindahkan posisi kursor digunakan fungsi setCursorPos.

Setelah memahami cara menampilkan karakter dengan warna karakternya, tentu kita akan bertanya bagaimana cara menampilkan *string* (rangkaian karakter) dengan warna-warna karakternya. Untuk menjawab pertanyaan ini marilah pelajari kode program berikut ini. Simpan project berikut ini dengan nama file contoh07.ide dan kode programnya dengan nama file contoh07.cpp.

contoh07.cpp:

```
01 #include <conio.h>
02 #include <dos.h>
03 #include <stdlib.h>
04
05 #define VIDEO_INT 0x10
06 #define UCHAR unsigned char
07
08 void getCursorPos(UCHAR *y, UCHAR *x);
09 void setCursorPos(UCHAR y, UCHAR x);
10 void writeChar(UCHAR letter, UCHAR attr);
11 void writeString(UCHAR *str, UCHAR attr);
12
13 int main(void)
14 {
15     UCHAR baris, kolom;
16
17     getCursorPos(&baris, &kolom);    // Baca posisi kursor
18     writeChar('>', 0x1f);          // Cetak karakter >
19     setCursorPos(baris, ++kolom); // Pindahkan kursor
20
21     writeString(" Mencetak String ", 0x4f);
22     getCursorPos(&baris, &kolom);
23     setCursorPos(baris, ++kolom);
24
25     writeChar('<', 0x1f);          // Cetak karakter <
26     setCursorPos(baris, ++kolom); // Pindahkan kursor
27     getch();
28
29     return EXIT_SUCCESS;
30 }
31
32 void getCursorPos(UCHAR *y, UCHAR *x) // Baca posisi
```

```

33 {                                     // kursor
34     UCHAR row, col;
35
36     asm mov ah, 0x03;      // Register AH = 3 heksadesimal
37     asm mov bh, 0x00;      // Register BH = 0 heksadesimal
38     asm int VIDEO_INT;   // Lakukan interupsi
39     asm mov row, dh;      // Salin register DH ke row
40     asm mov col, dl;      // Salin register DL ke col
41
42     *y = row; *x = col;    // Salin row ke y, col ke x
43
44     return;
45 }
46
47 void setCursorPos(UCHAR y, UCHAR x) // Memindahkan
48 {                                     // Posisi kursor
49     asm mov ah, 0x02;      // Register AH = 3 heksadesimal
50     asm mov bh, 0x00;      // Register BH = 0 heksadesimal
51     asm mov dh, y;        // Register DH = letak baris
52     asm mov dl, x;        // Register DL = letak kolom
53     asm int VIDEO_INT;   // Lakukan interupsi
54
55     return;
56 }
57
58 void writeChar(UCHAR letter, UCHAR attr) // Mencetak
59 {                                     // huruf
60     asm mov ah, 0x09;      // Register AH = 9 heksadesimal
61     asm mov al, letter;    // Register AL = hurufnya
62     asm mov bh, 0x00;      // Register BH = 0 heksadesimal
63     asm mov bl, attr;      // Register BL = warna huruf
64     asm mov ch, 0x00;      // Register CH dan CL menentukan
65     asm mov cl, 0x01;      // banyak pencetakan
66     asm int VIDEO_INT;   // Lakukan interupsi
67
68     return;
69 }
70
71 void writeString(UCHAR *str, UCHAR attr) // Mencetak
72 {                                     // string
73     UCHAR x, y;
74
75     getCursorPos(&y, &x);      // Simpan posisi kursor
76
77     for (; *str != '\0'; str++) // Loop sampai ditemukan
78     {                           // NULL
79         if (x > 79)           // Jika sudah sampai kolom
80         {                     // ke-80, pindah baris dan
81             y++; x = 0;       // pindah ke kolom ke-1
82         }
83
84         setCursorPos(y, x++); // Pindahkan posisi kursor
85         writeChar(*str, attr); // Cetak per karakter
86     }
87
88     return;
89 }

```

Program contoh07.cpp merupakan pengembangan dari program contoh06.cpp. Pada program contoh07.cpp terdapat fungsi writeString, fungsi ini menggunakan parameter str dan attr. Parameter str dikirimkan secara acuan dan berisi rangkaian karakter (string) yang akan dicetak. Sedangkan parameter attr dikirimkan secara nilai untuk menentukan warna string saat ditampilkan. Fungsi writeString memanggil fungsi writeChar untuk mencetak rangkaian karakternya satu per satu.

2.4. Membaca Karakter pada Posisi Kursor

Pada sub bab sebelumnya sudah dipelajari bagaimana cara menampilkan karakter dan string pada posisi kursor tertentu di layar. Pada sub bab ini akan dipelajari cara mengetahui nilai karakter dan warna karakter yang sudah tercetak dilayar. Seperti telah kita ketahui sebelumnya bahwa layar monitor pada mode teks normal terdiri dari suatu baris dan kolom. Dengan mengarahkan kursor pada baris dan kolom tertentu, dapat kita ketahui karakter/huruf yang tercetak dan warna dari karakter/huruf tersebut.

Nomor interupsi yang digunakan untuk mengetahui karakter dan warna karakter pada posisi tertentu adalah interupsi 10 heksadesimal servis 8 heksadesimal. Berikut ini adalah prosedur untuk menjalankan interupsi tersebut:

- Register AH harus bernilai 8 heksadesimal.
- Register BH berisi nomor halaman tampilan, halaman pertama nilainya 0.

Setelah interupsi dijalankan maka register AH akan berisi nilai warna dari karakter dan register AL akan berisi karakter/huruf yang ditampilkan. Berikut ini adalah contoh sederhana untuk membaca karakter pada posisi tertentu dilayar. Untuk menyederhanakan kode program, contoh berikut akan menggunakan fungsi standar gotoxy, textcolor, textbackground, cprintf dan int86. Simpan contoh program berikut ini dengan nama project contoh08.ide dan file kode program contoh08.cpp.

contoh08.cpp:

```
01 #include <conio.h>
02 #include <dos.h>
03 #include <stdio.h>
04 #include <stdlib.h>
05
06 #define VIDEO_INT 0x10          // Nomor interupsi video
07 #define UCHAR unsigned char    // Tipe data UCHAR
08
09 UCHAR getCharAttr(UCHAR *attr);
10
11 int main(void)
12 {
13     UCHAR huruf, warna;
14
15     clrscr();                  // Bersihkan layar
16     gotoxy(10, 5); textcolor(15); // Warna karakter
17     textbackground(5);         // Warna dasar karakter
18     cprintf(" Latihan C++ "); // Cetak string
19     gotoxy(13, 5);            // Pindah posisi kursor
20
21     huruf = getCharAttr(&warna); // Baca nilai karakter
22                                // dan atributnya
23
24     gotoxy(1, 7);
25     printf("Karakter pada baris 5 kolom 13: %c\n", huruf);
```

```

26     printf("Warna\\atribut dari karakter : %#x\n", warna);
27     getch();
29
30     return EXIT_SUCCESS;
31 }
32
33 UCHAR getCharAttr(UCHAR *attr) // Fungsi untuk membaca
34 {                                // karakter dan atributnya
35     union REGS in, out;           // pada posisi kursor
36
37     in.h.ah = 0x08;              // AH = 8 heksadesimal
38     in.h.bh = 0x00;              // BH = 0, halaman layar
39     int86(VIDEO_INT, &in, &out); // Lakukan interupsi
40
41     *attr = out.h.ah;            // Salin nilai AH di attr
42
43     return out.h.al;             // Kembalikan nilai AL
44 }

```

Pada program diatas, fungsi yang dibuat untuk membaca karakter dan warna atributnya adalah fungsi getCharAttr. Fungsi ini mengirimkan parameter dengan tipe data unsigned character secara acuan. Setelah fungsi tersebut dijalankan, parameter attr berisi nilai warna atribut dari karakter sedangkan fungsi getCharAttr sendiri mengembalikan nilai karakter yang dibaca.

2.5. ASCII Extended Character Set

ASCII Extended Character Set (set karakter ASCII perluasan) adalah karakter ASCII dengan kode atau nomor urut 128 sampai dengan 255 desimal. Umumnya set karakter perluasan ini digunakan agar tampilan program yang berbasis teks menjadi lebih menarik karena dapat digunakan untuk menampilkan bingkai, tabel, simbol-simbol khusus dalam aksara Yunani (seperti *alpha*, *beta*, *gamma* dan seterusnya) dan simbol-simbol khusus matematika (seperti integral, akar kuadrat dan pangkat dua). Gambar-gambar berikut ini menampilkan nomor urut serta karakter ASCII reguler dan perluasan.

Regular ASCII Chart (character codes 0 - 127)										
000 (nul)	016 ▶ (dle)	032 sp	048 0	064 @	080 P	096 ^	112 p			
001 ⓧ (soh)	017 ↵ (dc1)	033 !	049 1	065 A	081 Q	097 a	113 q			
002 ⓨ (stx)	018 ♫ (dc2)	034 "	050 2	066 B	082 R	098 b	114 r			
003 ♥ (etx)	019 !! (dc3)	035 #	051 3	067 C	083 S	099 c	115 s			
004 ♦ (eot)	020 ¶ (dc4)	036 \$	052 4	068 D	084 T	100 d	116 t			
005 ± (enu)	021 \$ (nak)	037 %	053 5	069 E	085 U	101 e	117 u			
006 ♣ (ack)	022 = (syn)	038 &	054 6	070 F	086 V	102 f	118 v			
007 • (bel)	023 ± (etb)	039 ^	055 7	071 G	087 W	103 g	119 w			
008 ☐ (bs)	024 ↑ (can)	040 <	056 8	072 H	088 X	104 h	120 x			
009 (tab)	025 ↓ (em)	041 >	057 9	073 I	089 Y	105 i	121 y			
010 (lf)	026 (eof)	042 *	058 :	074 J	090 Z	106 j	122 z			
011 ⓧ (vt)	027 + (esc)	043 +	059 ;	075 K	091 [107 k	123 t			
012 ♪ (np)	028 ↳ (fs)	044 ,	060 <	076 L	092 \	108 l	124 \			
013 (cr)	029 ♫ (gs)	045 -	061 =	077 M	093]	109 m	125]			
014 ⓧ (so)	030 ♪ (rs)	046 ,	062 >	078 N	094 _	110 n	126 _			
015 ⓧ (si)	031 ↴ (us)	047 /	063 ?	079 O	095 _	111 o	127 o			

Gambar II.1. Karakter ASCII reguler

Extended ASCII Chart (character codes 128 - 255)													
128	¢	143	À	158	฿	172	¤	186	₩	200	₪	214	₪
129	ú	144	É	159	ƒ	173	í	187	₪	201	₪	215	₪
130	é	145	æ	160	á	174	«	188	₪	202	₪	216	₪
131	à	146	Œ	161	í	175	»	189	₪	203	₪	217	₪
132	ä	147	å	162	ö	176	„	190	₪	204	₪	218	₪
133	ä	148	ö	163	ó	177	„	191	₪	205	=	219	₪
134	ä	149	ö	164	ñ	178	„	192	₪	206	₪	220	₪
135	ç	150	û	165	ñ	179	„	193	₪	207	₪	221	₪
136	ë	151	ù	166	ë	180	„	194	₪	208	₪	222	₪
137	ë	152	ÿ	167	ø	181	„	195	₪	209	₪	223	₪
138	ë	153	ø	168	ë	182	„	196	₪	210	₪	224	₪
139	í	154	ü	169	í	183	„	197	₪	211	₪	225	₪
140	í	155	¢	170	í	184	„	198	₪	212	₪	226	₪
141	í	156	£	171	½	185	„	199	₪	213	₪	227	₪
142	ñ	157	¥							214	₪	241	₪
										215	₪	228	₪
										216	₪	229	₪
										217	₪	230	₪
										218	₪	231	₪
										219	₪	232	₪
										220	₪	233	₪
										221	₪	234	₪
										222	₪	235	₪
										223	₪	236	₪
										224	₪	237	₪
										225	₪	238	₪
										226	₪	239	₪
										227	₪	240	₪
										228	₪	241	₪
										229	₪	242	₪
										230	₪	243	₪
										231	₪	244	₪
										232	₪	245	₪
										233	₪	246	₪
										234	₪	247	₪
										235	₪	248	₪
										236	₪	249	₪
										237	₪	250	₪
										238	₪	251	₪
										239	₪	252	₪
										240	₪	253	₪
										241	₪	254	₪
										242	₪	255	₪

Gambar II.2. Karakter ASCII perluasan (extended)

Berikut ini adalah contoh program yang sangat sederhana untuk membuat sebuah bingkai. Program ini akan menggunakan fungsi writeChar, writeString, setCursorPos yang telah dibuat pada program sebelumnya. Simpan program berikut ini dengan nama file project contoh09.ide dan nama file kode program contoh09.cpp.

contoh09.cpp:

```

001 #include <conio.h>
002 #include <dos.h>
003 #include <stdio.h>
004 #include <stdlib.h>
005
006 #define VIDEO_INT 0x10
007 #define UCHAR unsigned char
008
009 void setMode(UCHAR mode);
010 void getCursorPos(UCHAR *y, UCHAR *x);
011 void setCursorPos(UCHAR y, UCHAR x);
012 void writeChar(UCHAR letter, UCHAR attr);
013 void writeString(UCHAR *str, UCHAR attr);
014
015 int main(void)
016 {
017     UCHAR baris, kolom;
018     UCHAR pilih;
019
020     setMode(3);
021     setCursorPos(4, 4); writeChar(213, 0x17);
022     setCursorPos(4, 74); writeChar(184, 0x17);
023     setCursorPos(20, 4); writeChar(192, 0x17);
024     setCursorPos(20, 74); writeChar(217, 0x17);
025
026     for (baris = 5; baris < 20; baris++)
027     {
028         setCursorPos(baris, 4); writeChar(179, 0x17);
029         setCursorPos(baris, 74); writeChar(179, 0x17);
030     }
031
032     for (kolom = 5; kolom < 74; kolom++)
033     {

```

```

034     setCursorPos(4, kolom); writeChar(205, 0x17);
035     setCursorPos(20, kolom); writeChar(196, 0x17);
036 }
037
038 setCursorPos(4, 5); writeChar(181, 0x17);
039 setCursorPos(4, 6);
040 writeString("Bingkai dengan ASCII", 0x1f);
041 setCursorPos(4, 26); writeChar(198, 0x17);
042
043 for (baris = 5; baris < 20; baris++)
044 {
045     for (kolom = 5; kolom < 74; kolom++)
046     {
047         setCursorPos(baris, kolom);
048         writeChar(0x20, 0x1e);
049     }
050 }
051
052 setCursorPos(12, 25);
053 writeString("Akhiri program (Y/T)? [    ]", 0x1e);
054
055 for (;;)
056 {
057     setCursorPos(12, 49);
058     pilih = getch();
059
060     writeChar(pilih, 0x1e);
061
062     if ((pilih == 'Y') || (pilih == 'y'))
063         break;
064 }
065
066 return EXIT_SUCCESS;
067 }
068
069 void setMode(UCHAR mode) // Mengubah mode
070 {                         // tampilan layar
071     asm mov ah, 0x00;        // Register AH = 0
072     asm mov al, mode;       // Register AL = mode
073     asm int VIDEO_INT       // Lakukan interupsi
074
075     return;
076 }
077
078 void getCursorPos(UCHAR *y, UCHAR *x) // Baca posisi
079 {                                     // kursor
080     UCHAR row, col;
081
082     asm mov ah, 0x03;        // Register AH = 3 heksadesimal
083     asm mov bh, 0x00;        // Register BH = 0 heksadesimal
084     asm int VIDEO_INT;      // Lakukan interupsi
085     asm mov row, dh;        // Salin register DH ke row
086     asm mov col, dl;        // Salin register DL ke col
087
088     *y = row; *x = col;    // Salin row ke y, col ke x
089
090     return;

```

```

091 }
092
093 void setCursorPos( UCHAR y, UCHAR x) // Memindahkan
094 { // Posisi kurSOR
095     asm mov ah, 0x02; // Register AH = 3 heksadesimal
096     asm mov bh, 0x00; // Register BH = 0 heksadesimal
097     asm mov dh, y; // Register DH = letak baris
098     asm mov dl, x; // Register DL = letak kolom
099     asm int VIDEO_INT; // Lakukan interupsi
100
101     return;
102 }
103
104 void writeChar(UCHAR letter, UCHAR attr) // Mencetak
105 { // huruf
106     asm mov ah, 0x09; // Register AH = 9 heksadesimal
107     asm mov al, letter; // Register AL = hurufnya
108     asm mov bh, 0x00; // Register BH = 0 heksadesimal
109     asm mov bl, attr; // Register BL = warna huruf
110     asm mov ch, 0x00; // Register CH dan CL menentukan
111     asm mov cl, 0x01; // banyak pencetakan
112     asm int VIDEO_INT; // Lakukan interupsi
113
114     return;
115 }
116
117 void writeString(UCHAR *str, UCHAR attr) // Mencetak
118 { // string
119     UCHAR x, y;
120
121     getCursorPos(&y, &x); // Simpan posisi kurSOR
122
123     for (; *str != '\0'; str++) // Loop sampai ditemukan
124     { // NULL
125         if (x > 79)
126         {
127             y++; x = 0; // Jika sudah sampai kolom
128             // ke-80, pindah baris dan
129             // pindah ke kolom ke-1
130
131             setCursorPos(y, x++); // Pindahkan posisi kurSOR
132             writeChar(*str, attr); // Cetak per karakter
133
134     }
135 }

```

2.6. Membuat Class untuk Operasi Layar pada Modus Teks

Setelah memahami cara menggunakan fungsi int86 atau teknik *inline assembly* untuk melakukan operasi pada layar monitor, maka kita dapat membuat sebuah class tersendiri untuk melakukan operasi pada layar monitor. Class yang akan kita buat akan mengenkapsulasi fungsi-fungsi setMode, getMode, writeChar, writeString, setCursorPos, getCursorPos dan getCharAttr. Namun, agar class yang dibuat dalam proses eksekusi program (*runtime*) bekerja lebih optimal maka semua fungsi anggota (*method*) yang pada pembahasan sebelumnya dibuat menggunakan fungsi standar int86, kali ini akan ditulis ulang menggunakan teknik inline assembly.

Dalam pembuatan class dapat dilakukan tanpa membuat project terlebih dahulu, jadi cukup dilakukan dengan memilih menu **File > New > Text Edit** dari IDE Borland C++ 5.02. Class yang akan dibuat bernama Screen dan simpan kode program berikut ini dengan nama screen.cpp.

screen.cpp:

```
001 /*  
002     screen.cpp  
003     Class library untuk operasi layar monitor.  
004     Hak Cipta Pebi Yudha K.  
005     April 2009  
006  
007     Disusun sebagai contoh program  
008     Modul Praktikum Pemrograman C++ Lanjutan  
009     AMIK BSI - Jakarta  
010 */  
011  
012 #define UCHAR unsigned char  
013 #define VIDEO_INT 0x10  
014  
015 class Screen  
016 {  
017     private:  
018         UCHAR mode, row, col;  
019         UCHAR rownum, colnum;  
020         UCHAR active_page, visual_page;  
021         UCHAR attribute;  
022  
023     public:  
024         Screen(void); // Konstruktor default  
025         Screen(UCHAR mode);  
026  
027         ~Screen(void); // Desktruktor default  
028  
029         void setMode(UCHAR mode);  
030         UCHAR getMode(void);  
031  
032         void setCursorPos(UCHAR y, UCHAR x);  
033         void getCursorPos(UCHAR *y, UCHAR *x);  
034  
035         void writeChar(UCHAR letter);  
036         void writeString(UCHAR *str);  
037  
038         void setAttribute(UCHAR attr);  
039         UCHAR getCharAttr(UCHAR *attr);  
040  
041         void setActivePage(UCHAR page);  
042         UCHAR getPage(void);  
043  
044         void setVisualPage(UCHAR page);  
045         UCHAR getVisualPage(void);  
046  
047         void cls(void);  
048     }; // Akhir prototype class Screen  
049  
050 Screen::Screen(void) // Konstruktor default.  
051 {                                // M memset layar pada
```

```

052     UCHAR mode;           // mode 25 baris dan
053                     // 80 kolom, warna huruf
054     this->row = 0;        // abu-abu dan latar
055     this->col = 0;        // hitam, 16 warna.
056
057     this->mode = mode = 0x03;
058
059     this->rownum = 25;
060     this->colnum = 80;
061
062     this->active_page = 0;
063     this->visual_page = 0;
064
065     this->attribute = 7;
066
067     asm mov ah, 0x00;
068     asm mov al, mode;
069     asm or al, 0x80;
070     asm int VIDEO_INT;
071
072     return;
073 }
074
075 Screen::Screen(UCHAR mode) // Konstruktur
076 {                         // dengan
077     this->setMode(mode);    // parameter mode
078                     // layar.
079     return;
080 }
081
082 Screen::~Screen(void) // Destruktor default.
083 {                      // Tidak melakukan
084     return;              // apapun.
085 }
086
087 void Screen::setMode(UCHAR mode) // Memilih
088 {                         // mode
089     this->mode = mode;      // layar.
090
091     this->active_page = 0;
092     this->visual_page = 0;
093
094     this->rownum = 25;
095
096     switch (mode) // Menentukan banyak kolom
097     {             // berdasarkan nomor mode.
098         case 0x00 :
099             case 0x01 : this->colnum = 40;
100                 break;
101             case 0x02 :
102             case 0x03 :
103             case 0x07 : this->colnum = 80;
104                 break;
105
106             default   : this->colnum = 0;
107                 break;
108     }

```

```

109     asm mov ah, 0x00;
110     asm mov al, mode;
111     asm or al, 0x80;
112     asm int VIDEO_INT;
113
114     return;
115 }
116
117
118 UCHAR Screen::getMode(void) // Mengetahui
119 {                           // mode layar.
120     return this->mode;
121 }
122
123 void Screen::setCursorPos(UCHAR y, UCHAR x)
124 {
125     UCHAR page = this->active_page;
126
127     if ((y > this->rownum) ||
128         (x > this->colnum))
129     y = x = 0;
130
131     this->row = y;           // Menentukan posisi
132     this->col = x;          // kurSOR.
133
134     asm mov ah, 0x02;
135     asm mov bh, page;
136     asm mov dh, y;
137     asm mov dl, x;
138     asm int VIDEO_INT;
139
140     return;
141 }
142
143 void Screen::getCursorPos(UCHAR *y, UCHAR *x)
144 {
145     *y = this->row;        // Mengetahui posisi
146     *x = this->col;        // kurSOR.
147
148     return;
149 }
150
151 void Screen::writeChar(UCHAR letter)
152 {
153     UCHAR page = this->active_page;
154     UCHAR attr = this->attribute;
155
156     asm mov ah, 0x09;      // Menuliskan satu
157     asm mov al, letter;   // karakter dan
158     asm mov bh, page;    // atributnya
159     asm mov bl, attr;
160     asm mov ch, 0x00;
161     asm mov cl, 0x01;
162     asm int VIDEO_INT;
163
164     return;
165 }

```

```

166
167 void Screen::writeString(UCHAR *str)
168 {
169     for (; *str != '\0'; str++)
170     {
171         if (this->col > this->colnum)
172         {
173             this->row++;           // Mencetak rangkaian
174             this->col = 0;          // karakter (string).
175         }
176
177         this->setCursorPos(this->row,
178                             ++this->col);
179         this->writeChar(*str);
180     }
181
182     return;
183 }
184
185 void Screen::setAttribute(UCHAR attr)
186 {
187     this->attribute = attr;    // Mengubah warna
188                                // huruf dan warna
189     return;                   // latar.
190 }
191
192 UCHAR Screen::getCharAttr(UCHAR *attr)
193 {
194     UCHAR page = this->active_page;
195     UCHAR letter, attribute;
196
197     asm mov ah, 0x08;          // Mengetahui
198     asm mov bh, page;         // karakter dan
199     asm int VIDEO_INT;        // atributnya pada
200     asm mov attribute, ah;   // posisi kursor.
201     asm mov letter, al;
202
203     this->attribute = *attr = attribute;
204
205     return letter;
206 }
207
208 void Screen:: setActivePage(UCHAR page)
209 {
210     this->active_page = page; // Menentukan
211                                // halaman aktif.
212     return;
213 }
214
215 UCHAR Screen::getActivePage(void)
216 {
217     // Mengetahui
218     return this->active_page; // halaman aktif.
219 }
220
221 void Screen::setVisualPage(UCHAR page)
222 {
223     if (page > 7) page = 0;

```

```

223     this->visual_page = page;
225
226     asm mov ah, 0x05;      // Menentukan halaman
227     asm mov al, page;    // yang ditampilkan.
228     asm int VIDEO_INT;
229
230     return;
231 }
232
233 UCHAR Screen::getVisualPage(void)
234 {                               // Mengetahui
235     return this->visual_page; // halaman yang
236 }                               // ditampilkan.
237
238 void Screen::cls(void)
239 {                               // Membersihkan
240     UCHAR mode = this->mode; // tampilan layar
241                               // serta
242     this->row = 0;           // memposisikan
243     this->col = 0;          // kursor di 0, 0.
244
245     asm mov ah, 0x00;
246     asm mov al, mode;
247     asm int VIDEO_INT;
248
249     return;
250 }

```

Kode program screen.cpp diatas bukanlah suatu program untuk dikompilasi dan dijalankan, namun sebuah pustaka class (*class library*) yang fungsinya hampir sama dengan penggunaan file header (.h). Pada kode program screen.cpp terdapat beberapa fungsi anggota dari class Screen yang sebelumnya tidak dibahas, yaitu setAttribute, setActivePage, getActivePage, setVisualPage, getVisualPage dan cls.

Fungsi anggota setAttribute digunakan untuk menentukan warna huruf dan warna latar (*background*). Fungsi ini mengirimkan parameter bertipe unsigned character yang menunjukkan warna huruf dan latar yang akan ditampilkan. Sedangkan untuk menentukan halaman halaman aktif digunakan fungsi setActivePage. Apa maksudnya halaman aktif? Halaman aktif adalah halaman memori video dimana proses operasi pada layar dilakukan, seperti memindahkan posisi kursor, menampilkan karakter, membersihkan layar dan sebagainya. Pada mode layar teks 80 baris 25 kolom terdapat delapan halaman memori video (halaman layar), yaitu halaman 0 sampai dengan halaman 7. Fungsi setActivePage digunakan untuk memilih nomor halaman tersebut dengan cara mengirimkan angka 0, 1, 2 sampai dengan 7 sebagai parameter. Umumnya pada saat pertama kali dijalankan program akan menggunakan halaman layar pertama (halaman 0) sebagai halaman visual (halaman dimana output program ditampilkan). Untuk mengetahui nomor halaman aktif digunakan fungsi anggota getActivePage yang mengembalikan nomor halaman aktif yang sedang digunakan.

Selain fungsi setActivePage dan getActivePage untuk menangani nomor halaman layar, class Screen juga memiliki fungsi anggota untuk mengatur halaman visual. Apa maksudnya halaman visual? Halaman visual adalah nomor halaman dimana output program ditampilkan dan dapat terlihat oleh pengguna. Lalu apakah perbedaan antara halaman aktif dengan halaman visual? Suatu program dapat saja menyembunyikan proses penampilan output dengan cara menuliskan karakter-karakter pada halaman layar pertama (halaman 0) tetapi halaman visual yang ditampilkan adalah halaman kedua (halaman 1). Karakter-karakter

yang dituliskan pada halaman 0 tersebut tidak akan benar-benar muncul di layar monitor sampai pengguna memrogram untuk memilih halaman 0 sebagai halaman visual. Jika tidak, maka halaman 1 tetap akan ditampilkan walaupun halaman 1 kosong. Untuk memilih halaman visual digunakan fungsi setVisualPage dan untuk mengetahui nomor halaman yang sedang dijadikan halaman visual digunakan fungsi getVisualPage.

Kegunaan fungsi anggota cls sendiri dapat diketahui dari namanya, yaitu untuk membersihkan tampilan layar (*clear screen*) dan memposisikan kursor pada baris 0 kolom 0 (baris pertama kolom pertama). Project contoh10.ide berikut ini adalah demonstrasi penggunaan pustaka screen.cpp. Maksud dari programnya adalah untuk memberikan contoh penggunaan fungsi anggota setActivePage dan setVisualPage yang merupakan fungsi anggota dari objek Screen. Agar project contoh10.ide ini berjalan maka salinlah file screen.cpp dalam direktori yang sama tempat Anda menyimpan project.

contoh10.cpp:

```
01 #include <dos.h>
02 #include <stdlib.h>
03 #include "screen.cpp"    // Header screen.cpp
04
05 int main(void)
06 {
07     UCHAR i, j;
08     Screen *layar = new Screen();
09
10     layar->setAttribute(0x9f);
11     layar->setActivePage(0);
12     layar->writeString("Halaman pertama");
13     layar->setAttribute(0xcf);
14     layar->setActivePage(1);
15     layar->writeString("Halaman ke dua");
16
17     for (i = 1; i < 11; i++)
18     {
19         j = i % 2;
20         layar->setVisualPage(j);
21         delay(3000);
22     }
23
24
25     delete layar;
26     return EXIT_SUCCESS;
27 }
```

Ketika program diatas dijalankan maka muncul tulisan "Halaman pertama" dan "Halaman ke dua" secara bergantian dengan jeda waktu selama tiga detik. Tulisan "Halaman pertama" ditampilkan pada halaman 0 sedangkan "Halaman ke dua" pada halaman 1.

2.7. Latihan-latihan Bab II

1. Jelaskan sintaks dan parameter fungsi standar int86!
2. Apa yang dimaksud dengan teknik inline assembly dan bagaimana cara menggunakannya pada kompilator Borland C++ 5.02?
3. Jelaskan kegunaan karakter ASCII dalam pembuatan program berbasis teks!
4. Buat sebuah program dengan menggunakan int86 untuk menampilkan output seperti tulisan dibawah ini dengan warna huruf putih dan warna latar biru!

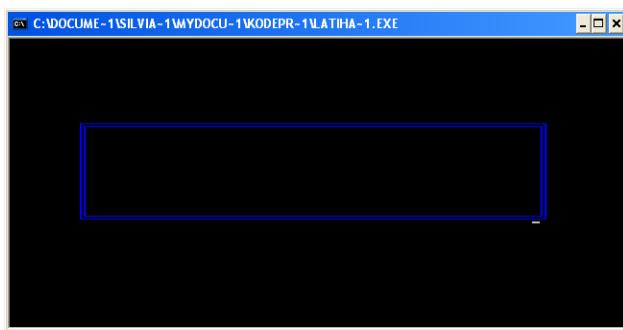
A

A B
A B C
A B C D

5. Buatlah sebuah program untuk menampilkan output seperti dibawah ini menggunakan teknik inline assembly!

A B C D E
A B C D
A B C
A B
A

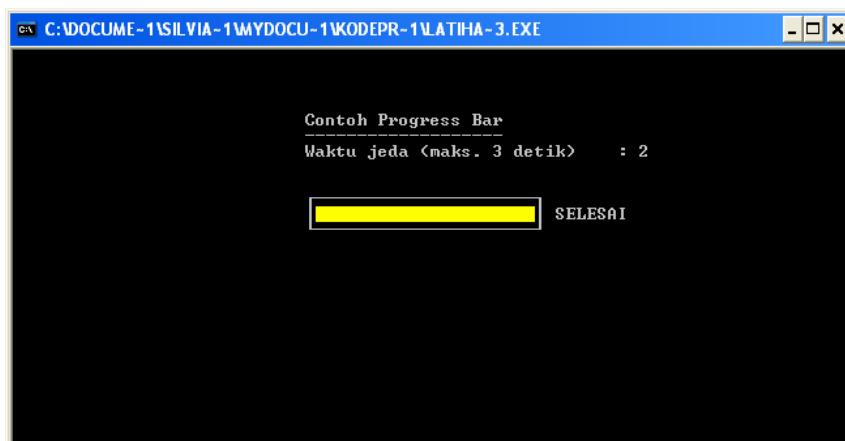
6. Dengan memanfaatkan pustaka class screen.cpp buatlah sebuah program dengan output seperti dibawah ini!



7. Dengan memanfaatkan pustaka class screen.cpp buatlah sebuah program dengan ketentuan sebagai berikut:

- ❑ Pengguna menginputkan sebuah string pada halaman layar 0 (halaman pertama).
- ❑ String yang diketikan oleh pengguna akan ditampilkan setelah jeda 5 detik di halaman layar 1 (halaman kedua).
- ❑ Setelah jeda 5 detik di halaman 1, program akan kembali menampilkan halaman 0 kemudian membersihkan layar.
- ❑ Setelah jeda 5 detik di halaman 0, program kembali ke halaman 1 dan jeda kembali selama 3 detik kemudian program berhenti.

8. Dengan memanfaatkan pustaka class screen.cpp buatlah sebuah program dengan rancangan input dan output seperti dibawah ini!



BAB III

Input Menggunakan Keyboard

3.1. Interupsi BIOS untuk Input Melalui Keyboard

Keyboard atau papan ketik merupakan alat input utama pada sebuah PC. Umumnya seseorang yang sedang mempelajari bahasa pemrograman C/C++ akan memanfaatkan fungsi-fungsi standar seperti scanf, getch, getchar, getche, gets dan cin untuk bisa melakukan input melalui keyboard, maka pada bab ini akan dipelajari bagaimana fungsi-fungsi tersebut bekerja.

BIOS menyediakan sebuah nomor interupsi khusus yang digunakan untuk melakukan input melalui keyboard, yaitu interupsi 16 heksadesimal. Operasi-operasi pada keyboard dapat ditentukan melalui nomor service yang disimpan pada register AH.

3.2. Memasukan Satu Karakter dan Mendeteksi Penekanan Tombol Extended

Interupsi 16 heksadesimal servis 0 heksadesimal adalah nomor interupsi yang dapat digunakan untuk melakukan input satu karakter melalui keyboard. Fungsi standar getch, getchar, dan getche memanfaatkan interupsi ini. Berikut ini adalah prosedur yang harus dilakukan untuk melakukan interupsi 16 heksadesimal servis 0 heksadesimal:

- ❑ Register AH harus bernilai 0.
- ❑ Jalankan interupsi 16 heksadesimal.

Setelah Interupsi dijalankan maka:

- ❑ Register AH akan berisi dengan kode scan papan ketik.
- ❑ Register AL berisi kode ASCII dari karakter yang diketikan atau bernilai 0 jika tombol keyboard yang ditekan adalah tombol *extended*.

Dari penjelasan diatas, ada beberapa hal lagi yang harus lebih diperjelas. Karakter yang diketikan melalui keyboard tidak akan ditampilkan pada layar monitor dan posisi kursor tidak akan berpindah. Tombol keyboard yang diketikan dapat diketahui nilainya pada register AL, jika register AL bernilai nol, maka dapat disimpulkan bahwa tombol keyboard yang ditekan adalah tombol *extended* (tombol-tombol khusus).

Tombol extended atau tombol khusus adalah tombol yang disediakan untuk maksud tertentu dari sebuah program. Contoh tombol-tombol khusus adalah tombol CTRL, ALT, SHIFT, INS, F1, F2 dan sebagainya. Untuk mengetahui tombol khusus apa yang ditekan ketika register AL bernilai nol, dapat diketahui dari nilai yang disimpan pada register AH. Nilai pada register AH adalah *keyboard scan code* yang memberitahukan tombol khusus apa yang ditekan.

Berikut ini adalah contoh program yang menunjukkan cara melakukan input satu karakter menggunakan interupsi 16 heksadesimal servis 0 yang ditulis dengan teknik inline assembly dan memanfaatkan pustaka class screen.cpp yang telah dibuat pada bab sebelumnya. Buatlah sebuah project seperti yang dilakukan pada bab-bab sebelumnya, simpan project tersebut dengan nama file contoh11.ide dengan file kode program contoh11.cpp pada direktori yang sama dengan file screen.cpp

contoh11.cpp:

```
01 #include <stdlib.h>
```

```

02 #include "screen.cpp"
03
04 #define KEY_INT 0x16      /* Nomor interupsi keyboard */
05
06 UCHAR getKey(void);
07
08 int main(void)
09 {
10     Screen *layar = new Screen();
11     UCHAR karakter;
12
13     latar->setMode(0x03);
14     latar->setCursorPos(4, 14);
15     latar->writeString("Input satu karakter:");
16     latar->setCursorPos(4, 34);
17
18     karakter = getKey();
19
20     latar->setCursorPos(5, 14);
21     latar->writeString("Karakter yang anda ketik adalah");
22     latar->setCursorPos(5, 46);
23     latar->writeChar(karakter);
24     latar->setCursorPos(6, 14);
25     latar->writeString("Tekan sembarang tombol ...");
26
27     getKey();
28
29     delete latar;
30     return EXIT_SUCCESS;
31 }
32
33 UCHAR getKey(void)
34 {
35     UCHAR key;
36
37     asm mov ah, 0x00;    /* Register AH = 0          */
38     asm int KEY_INT;    /* Lakukan interupsi        */
39     asm mov key, al;    /* Salin nilai register AH ke key */
40
41     return key;
42 }

```

Pada kode program contoh11.cpp fungsi getKey adalah fungsi yang digunakan untuk input satu karakter. Fungsi getKey tidak memerlukan parameter dan mengembalikan nilai bertipe unsigned character (didefinisikan dalam file screen.cpp sebagai UCHAR). Nilai yang dikembalikan oleh fungsi ini adalah karakter yang diketikan pada tombol keyboard. Fungsi ini sama dengan fungsi standar getch.

Setelah mempelajari cara memasukan satu karakter, maka langkah berikutnya adalah mempelajari cara mendeteksi penekanan tombol-tombol extended. Project contoh12.ide berikut ini memberikan contoh cara mendeteksi penekanan tombol F1, F2, F3, dan F4. Simpan kode program contoh12.cpp berikut ini pada direktori yang sama dengan file screen.cpp.

contoh12.cpp:

```
01 #include <stdlib.h>
```

```

02 #include "screen.cpp"
03
04 #define KEY_INT 0x16 /* Nomor interupsi keyboard */
05 #define KEY_F1 0x3b /* Kode scan tombol F1 */
06 #define KEY_F2 0x3C /* Kode scan tombol F2 */
07 #define KEY_F3 0x3d /* Kode scan tombol F3 */
08 #define KEY_F4 0x3e /* Kode scan tombol F4 */
09
10 #define TRUE 1
11
12 UCHAR getSpecialKey(void);
13
14 int main(void)
15 {
16     Screen *layar = new Screen();
17     UCHAR tombol;
18
19     layar->setMode(0x03);
20     layar->setCursorPos(5, 9);
21     layar->writeString("Tekan F1, F2, F3");
22     layar->setCursorPos(6, 9);
23     layar->writeString("Tekan F4 untuk berhenti");
24     layar->setCursorPos(8, 9);
25
26     while (TRUE)
27     {
28         tombol = getSpecialKey(); // Periksa tombol yang ditekan
29
30         layar->setCursorPos(8, 9);
31
32         switch (tombol)
33         {
34             case KEY_F1 :
35                 layar->writeString("Tombol F1 ditekan");
36                 break;
37             case KEY_F2 :
38                 layar->writeString("Tombol F2 ditekan");
39                 break;
40             case KEY_F3 :
41                 layar->writeString("Tombol F3 ditekan");
42                 break;
43             case KEY_F4 :
44                 delete layar;
45                 return EXIT_SUCCESS;
46             default :
47                 layar->writeString("Tombol lain      ");
48                 break;
49         }
50     }
51 }
52
53 UCHAR getSpecialKey(void)
54 {
55     UCHAR key;
56
57     asm mov ah, 0x00; /* Register AH = 0 */
58     asm int KEY_INT; /* Lakukan interupsi 16 heksadesimal */

```

```

59     asm mov key, ah;      /* Salin nilai di AH ke variabel key */
60
61     return key;
62 }
```

Pada kode program contoh12.cpp fungsi getSpecialKey adalah fungsi yang digunakan untuk mendeteksi penekanan tombol-tombol khusus. Perhatikanlah pada baris kelima sampai baris kedelapan, karena setiap tombol khusus memiliki kode tersendiri maka pada baris kelima sampai kedelapan dideklarasikan konstanta untuk tombol F1 sampai F4. Kode scan untuk tombol F1 adalah 3b heksadesimal, F2 adalah 3c heksadesimal, dan seterusnya. Bandingkanlah kode program contoh11.cpp dengan kode program contoh12.cpp pada fungsi getKey dan getSpecialKey. Pada fungsi getKey nilai yang dijadikan sebagai nilai kembali fungsi (*return value*) adalah nilai yang tersimpan di register AL, sedangkan pada fungsi getSpecialKey nilai yang dikembalikan adalah nilai yang tersimpan di register AL.

3.3. Memasukan String Menggunakan Keyboard

Setelah memahami cara memasukan satu karakter melalui keyboard, maka pada sub bab ini akan dipelajari cara memasukan string menggunakan keyboard. Fungsi getKey dan getSpecialKey langsung mengembalikan nilai dari tombol yang ditekan tanpa menunggu penekanan tombol ENTER dan tidak menggeser posisi kursor ke sebelah kanan. Pada fungsi untuk memasukan rangkaian karakter berikut ini akan digunakan pustaka class screen.cpp untuk mengetahui letak kursor, memindahkan posisi kursor dan mencetak karakter.

Beberapa hal lain yang harus diperhatikan pada fungsi untuk memasukan string adalah sebagai berikut:

1. Tombol-tombol khusus seperti Esc, F1 s.d F12, Fn, CTRL, ALT, Page Up, Page Down, tombol tanda panah, Home, End, Ins, Del dan Break tidak dapat ditampilkan dan tidak dapat digunakan.
2. Tombol backspace akan menghapus satu karakter disebelah kiri, jika karakter pertama sudah terhapus maka tombol tidak berfungsi.
3. Karakter yang bisa dimasukan adalah karakter alphanumerik, simbol-simbol dan spasi. Karakter tab akan dianggap sebagai spasi.
4. Jumlah karakter yang dapat dimasukan harus dibatasi. Tombol ENTER tidak akan disimpan dan digunakan sebagai tanda bahwa string telah selesai dimasukan.
5. String pada bahasa pemrograman C/C++ adalah rangkaian karakter dalam bentuk ASCIIIZ (*null terminated string*).

Setelah memahami ide dasar memasukan rangkaian karakter, project contoh13.ide berikut ini akan memberikan contoh bagaimana menerapkan ide tersebut menggunakan teknik inline assembly. Project ini akan menggunakan pustaka class screen.cpp, jadi simpanlah kode program contoh13.cpp pada direktori yang sama dengan screen.cpp.

contoh13.cpp:

```

01 #include <stdlib.h>
02 #include "screen.cpp"
03
04 #define KEY_INT          0x16    /* Nomor interupsi keyboard */
05 #define KEY_BACKSPACE    0x08    /* Tombol Backspace */
06 #define KEY_RETURN        0x0d    /* Tombol Enter */
07 #define KEY_TAB           0x09    /* Tombol Tab */
08 #define KEY_SPACE         0x20    /* Tombol spasi */
09
10 #define NULL             0x00    /* ASCII 0 */
11 #define TRUE             1
```

```

12
13 UCHAR *getString(Screen *scr, UCHAR *str, UCHAR max);
14
15 int main(void)
16 {
17     Screen *layar = new Screen();
18     UCHAR str1[30];
19     UCHAR *str2;
20
21     layar->setMode(0x03);
22     layar->setCursorPos(5, 9);
23     layar->writeString("Ketikan sebuah string:");
24     layar->setCursorPos(5, 32);
25
26     str2 = getString(layar, str1, 25);
27
28     layar->setCursorPos(6, 9);
29     layar->writeString("String yang Anda ketikan adalah");
30     layar->setCursorPos(6, 41);
31     layar->writeString(str1);
32
33     /* getKey */
34     asm mov ah, 0x00;
35     asm int KEY_INT;
36
37     delete layar; return EXIT_SUCCESS;
38 }
39
40 UCHAR *getString(Screen *scr, UCHAR *str, UCHAR max)
41 {
42     UCHAR key = 0;
43     UCHAR i, x, y;
44
45     i = 0;
46
47     while (TRUE)
48     {
49         asm mov ah, 0x00;
50         asm int KEY_INT;
51         asm mov key, al;
52
53         if ((key == KEY_BACKSPACE) && (i > 0))
54         {
55             scr->getCursorPos(&y, &x);
56             scr->setCursorPos(y, --x);
57             scr->writeChar(KEY_SPACE);
58
59             *(str + i) = NULL; i--;
60         }
61
62         if ((key >= 32) && (key <= 126) && (i < max))
63         {
64             scr->getCursorPos(&y, &x);
65             scr->writeChar(key);
66             scr->setCursorPos(y, ++x);
67
68             *(str + i) = key; i++;

```

```

69     }
70
71     if ((key == KEY_TAB) && (i < max))
72     {
73         scr->getCursorPos(&y, &x);
74         scr->writeChar(KEY_SPACE);
75         scr->setCursorPos(y, ++x);
76
77         *(str + i) = KEY_SPACE; i++;
78     }
79
80     if (key == KEY_RETURN)
81     {
82         *(str + i) = NULL;
83         break;
84     }
85
86     if (i == max) *(str + i) = NULL;
87 }
88
89 return str;
90 }
```

Pada program contoh13.cpp, fungsi yang digunakan untuk memasukan string adalah fungsi getString. Fungsi getString menerima tiga parameter, yaitu scr dengan tipe data Screen yang dikirimkan secara referensi, str dengan tipe data unsigned character yang dikirimkan secara referensi dan parameter max dengan tipe data unsigned character yang dikirimkan secara nilai. Parameter scr digunakan untuk mengetahui posisi kursor dan memindahkan posisi kursor pada mode layar yang digunakan serta menampilkan karakter yang diketikan melalui keyboard. Parameter str adalah pointer karakter yang digunakan untuk menunjukan alamat memori dimana karakter-karakter yang dimasukan melalui keyboard disimpan dalam memori. Sedangkan parameter max digunakan untuk menentukan banyaknya karakter yang bisa dimasukan.

3.4. Memasukan String Berupa Kata Sandi

Pada aplikasi-aplikasi tertentu yang membutuhkan autentikasi seperti nama pengguna dan kata sandi, biasanya saat memasukan kata sandi akan ditampilkan karakter asteriks (*) sebagai ganti karakter yang diketikan. Hal ini dimaksudkan agar orang lain yang tidak berhak tidak bisa mencuri lihat kata sandi atau password yang sedang dimasukan.

Untuk membuat input password seperti yang telah dijelaskan diatas, kita dapat memodifikasi fungsi getString pada program contoh13.cpp. Project contoh14.ide berikut ini adalah contoh penerapan ide untuk memasukan password. Project ini akan menggunakan pustaka class screen.cpp, jadi simpanlah kode program contoh14.cpp pada direktori yang sama dengan file screen.cpp.

contoh14.cpp:

```

001 #include <stdlib.h>
002 #include "screen.cpp"
003
004 #define KEY_INT          0x16 /* Nomor interupsi keyboard */
005 #define KEY_BACKSPACE    0x08 /* Tombol Backspace */
006 #define KEY_RETURN        0x0d /* Tombol Enter */
007 #define KEY_TAB           0x09 /* Tombol Tab */
008 #define KEY_SPACE          0x20 /* Tombol spasi */
```

```

009
010 #define NULL          0x00 /* ASCII 0 */ *
011 #define TRUE           1
012
013 UCHAR *getString(Screen *scr, UCHAR *str, UCHAR max);
014 UCHAR *getPwdString(Screen *scr, UCHAR *pwd, UCHAR max);
015
016 int main(void)
017 {
018     Screen *layar = new Screen();
019     UCHAR *nama, nama2[16];
020     UCHAR *sandi, sandi2[21];
021     UCHAR kar = '*';
022
023     layar->setMode(0x03);
024     layar->setCursorPos(5, 9);
025     layar->writeString("Username:");
026     layar->setCursorPos(5, 19);
027
028     /* Memasukan username */
029     nama = getString(layar, nama2, 15);
030
031     layar->setCursorPos(6, 9);
032     layar->writeString("Password:");
033     layar->setCursorPos(6, 19);
034
035     /* Memasukan password */
036     sandi = getPwdString(layar, sandi2, 20);
037
038     layar->setCursorPos(8, 9);
039     layar->writeString("Selamat datang");
040     layar->setCursorPos(8, 24);
041     layar->writeString(nama);
042     layar->setCursorPos(9, 9);
043     layar->writeString("Password Anda adalah");
044     layar->setCursorPos(9, 30);
045     layar->writeString(sandi);
046
047     /* getKey */
048     asm mov ah, 0x00;
049     asm int KEY_INT;
050
051     delete layar; return EXIT_SUCCESS;
052 }
053
054 UCHAR *getString(Screen *scr, UCHAR *str, UCHAR max)
055 {
056     UCHAR key, i, x, y;
057
058     key = i = 0;
059
060     while (TRUE)
061     {
062         asm mov ah, 0x00;
063         asm int KEY_INT;
064         asm mov key, al;
065

```

```

066     if ((key == KEY_BACKSPACE) && (i > 0))
067     {
068         scr->getCursorPos(&y, &x);
069         scr->setCursorPos(y, --x);
070         scr->writeChar(KEY_SPACE);
071
072         *(str + i) = NULL; i--;
073     }
074
075     if ((key >= 32) && (key <= 126) && (i < max))
076     {
077         scr->getCursorPos(&y, &x);
078         scr->writeChar(key);
079         scr->setCursorPos(y, ++x);
080
081         *(str + i) = key; i++;
082     }
083
084     if ((key == KEY_TAB) && (i < max))
085     {
086         scr->getCursorPos(&y, &x);
087         scr->writeChar(KEY_SPACE);
088         scr->setCursorPos(y, ++x);
089
090         *(str + i) = KEY_SPACE; i++;
091     }
092
093     if (key == KEY_RETURN)
094     {
095         *(str + i) = NULL;
096         break;
097     }
098
099     if (i == max) *(str + i) = NULL;
100 }
101
102     return str;
103 }
104
105 UCHAR *getPwdString(Screen *scr, UCHAR *pwd, UCHAR max)
106 {
107     UCHAR key, i, x, y;
108
109     key = i = 0;
110
111     while (TRUE)
112     {
113         asm mov ah, 0x00;
114         asm int KEY_INT;
115         asm mov key, al;
116
117         if ((key == KEY_BACKSPACE) && (i > 0))
118         {
119             scr->getCursorPos(&y, &x);
120             scr->setCursorPos(y, --x);
121             scr->writeChar(KEY_SPACE);
122

```

```

123         *(pwd + i) = NULL; i--;
124     }
125
126     if ((key >= 32) && (key <= 126) && (i < max))
127     {
128         scr->getCursorPos(&y, &x);
129         scr->writeChar('*');
130         scr->setCursorPos(y, ++x);
131
132         *(pwd + i) = key; i++;
133     }
134
135     if ((key == KEY_TAB) && (i < max))
136     {
137         scr->getCursorPos(&y, &x);
138         scr->writeChar('*');
139         scr->setCursorPos(y, ++x);
140
141         *(pwd + i) = '*'; i++;
142     }
143
144     if (key == KEY_RETURN)
145     {
146         *(pwd + i) = NULL;
147         break;
148     }
149
150     if (i == max) *(pwd + i) = NULL;
151 }
152
153     return pwd;
154 }
```

Fungsi getPwdString pada program contoh14.cpp diatas adalah fungsi yang digunakan untuk memasukan string password. Perhatikanlah pada baris 129 dan 138! Pada baris 129 ketika ada karakter alphanumerik dan simbol yang diketikan maka yang akan ditampilkan adalah sebuah asteriks, demikian juga ketika tombol tab yang ditekan. Perbedaan lain antara fungsi getString dan getPwdString adalah karakter tab akan tetap disimpan sebagai tab (ASCII 9) bukan sebagai spasi.

3.5. Mengetahui Status Tombol On/Off

Pada keyboard ada tombol-tombol yang memiliki dua keadaan (on atau off) seperti Caps Lock, Num Lock, Scroll Lock, serta tombol yang memiliki status tekan (pressed/ditekan atau depressed/dilepas) seperti Alt, Ctrl, Ins, dan Shift. BIOS pada PC menyediakan interupsi 16 heksadesimal servis 2 untuk mengetahui status tombol-tombol tersebut. Berikut ini adalah prosedur untuk menjalankan interupsi 16 heksadesimal servis 2.

- Register AH harus bernilai 2 heksadesimal.
- Jalankan interupsi 16 heksadesimal.

Setelah interupsi dijalankan, register AL akan berisi nilai *BIOS keyboard flag* dengan ketentuan sebagai berikut:

- ⓐ Jika bit 0 bernilai 1 berarti tombol Shift kanan dilepas.
- ⓐ Jika bit 1 bernilai 1 berarti tombol Shift kiri dilepas.
- ⓐ Jika bit 2 bernilai 1 berarti tombol Ctrl dilepas.

- ⌚ Jika bit 3 bernilai 1 berarti tombol Alt dilepas.
- ⌚ Jika bit 4 bernilai 1 berarti tombol Scroll Lock on.
- ⌚ Jika bit 5 bernilai 1 berarti tombol Num Lock On.
- ⌚ Jika bit 6 bernilai 1 berarti tombol Caps Lock on.
- ⌚ Jika bit 7 bernilai 1 berarti tombol Ins aktif.

Berikut ini adalah contoh program untuk mengetahui cara menggunakan interupsi 16 heksadesimal servis 2. Simpan project ini dengan nama contoh15.ide pada direktori yang sama dengan file screen.cpp.

contoh15.cpp:

```

001 #include <stdlib.h>
002 #include "screen.cpp"
003
004 #define KEY_INT      0x16
005 #define KEY_ESC      0x1b
006
007 #define STATE_RSHIFT 0x01
008 #define STATE_LSHIFT 0x02
009 #define STATE_CTRL   0x04
010 #define STATE_ALT    0x08
011 #define STATE_SCROLL 0x10
012 #define STATE_NUM    0x20
013 #define STATE_CAPS   0x40
014 #define STATE_INS    0x80
015
016 #define TRUE        1
017
018 UCHAR getKeyState(UCHAR key);
019
020 int main(void)
021 {
022     Screen *layar = new Screen();
023     UCHAR tombol;
024
025     layar->setMode(0x03);
026     layar->setCursorPos(4, 14);
027     layar->writeString("SHIFT KANAN :");
028     layar->setCursorPos(5, 14);
029     layar->writeString("SHIFT KIRI   :");
030     layar->setCursorPos(6, 14);
031     layar->writeString("CTRL          :");
032     layar->setCursorPos(7, 14);
033     layar->writeString("ALT           :");
034     layar->setCursorPos(8, 14);
035     layar->writeString("SCROLL LOCK  :");
036     layar->setCursorPos(9, 14);
037     layar->writeString("NUM LOCK    :");
038     layar->setCursorPos(10, 14);
039     layar->writeString("CAPS LOCK   :");
040     layar->setCursorPos(11, 14);
041     layar->writeString("INS - Hentikan Program");
042
043     while (TRUE)
044     {
045         asm mov ah, 0x01; /* Menyembunyikan kursor

```

```

046     asm mov ch, 0x20; /* Register CX = 2000 heksadesimal */
047     asm mov cl, 0x00;
048     asm int 0x10;      /* BIOS Video Interupt Service 1 */
049
050     layar->setCursorPos(4, 30);
051
052     if (getKeyState(STATE_RSHIFT))
053         layar->writeString("DITEKAN");
054     else
055         layar->writeString("DILEPAS");
056
057     layar->setCursorPos(5, 30);
058
059     if (getKeyState(STATE_LSHIFT))
060         layar->writeString("DITEKAN");
061     else
062         layar->writeString("DILEPAS");
063
064     layar->setCursorPos(6, 30);
065
066     if (getKeyState(STATE_CTRL))
067         layar->writeString("DITEKAN");
068     else
069         layar->writeString("DILEPAS");
070
071     layar->setCursorPos(7, 30);
072
073     if (getKeyState(STATE_ALT))
074         layar->writeString("DITEKAN");
075     else
076         layar->writeString("DILEPAS");
077
078     layar->setCursorPos(8, 30);
079
080     if (getKeyState(STATE_SCROLL))
081         layar->writeString("ON ");
082     else
083         layar->writeString("OFF");
084
085     layar->setCursorPos(9, 30);
086
087     if (getKeyState(STATE_NUM))
088         layar->writeString("ON ");
089     else
090         layar->writeString("OFF");
091
092     layar->setCursorPos(10, 30);
093
094     if (getKeyState(STATE_CAPS))
095         layar->writeString("ON ");
096     else
097         layar->writeString("OFF");
098
099     layar->setCursorPos(11, 14);
100
101     if (getKeyState(STATE_INS)) break;
102 }
```

```

103     delete layar;
104     return EXIT_SUCCESS;
105 }
106 }
107
108 UCHAR getKeyState(UCHAR key)
109 {
110     UCHAR state;
111
112     asm mov ah, 0x02;    /* Register AH = 2           */
113     asm int KEY_INT;    /* Lakukan interupsi 16 heksadesimal */
114     asm and al, key;   /* AND-kan register AL sesuai tombol */
115     asm mov state, al; /* Salin nilai register AL ke state */
116
117     return state;
118 }

```

Pada program contoh15.cpp, fungsi getKeyState adalah fungsi yang digunakan untuk mengetahui status tombol. Perhatikanlah baris 7 sampai baris 14! Pada baris ketujuh sampai keempat belas dideklarasikan konstanta STATE_RSHIFT dan STATE_INS. Konstanta-konstanta ini digunakan untuk mengetahui status penekanan tombol Shift kanan sampai tombol Ins. Konstanta-konstanta inilah yang dikirimkan sebagai parameter pada fungsi getKeyState. Pada baris 114, setelah interupsi dijalankan nilai register AL di-AND-kan dengan nilai yang dikirimkan parameter fungsi getKeyState. Sebagai contoh, untuk mengetahui status tombol Shift kanan sedang ditekan atau dilepas, ditentukan dari nilai bit pertama pada register AL. Untuk mengetahui apakah nilai bit pertama 1 atau 0, maka nilai pada register AL harus di-AND-kan dengan nilai 0x01 atau 00000001 binari atau nilai konstanta STATE_RSHIFT. Kemudian nilai register AL yang telah di-AND-kan dijadikan nilai kembali fungsi getKeyState.

3.6. Konversi Nilai String Menjadi Numerik atau Sebaliknya

Semua nilai yang dimasukan melalui keyboard pada dasarnya adalah karakter atau rangkaian karakter (string). Nilai numerik yang dimasukan menggunakan fungsi standar scanf, cscanf, fscanf, gets dan cin pada dasarnya adalah nilai string yang telah mengalami proses konversi dari string menjadi bilangan bulat *integer* atau bilangan *real (floating point dan double)*. Agar kita bisa menggunakan fungsi getString yang telah kita buat untuk memasukan data numerik, maka string yang dimasukan melalui fungsi getString harus dikonversi menjadi numerik. Begitu juga sebaliknya, untuk menampilkan data numerik menggunakan fungsi anggota writeString dari class Screen, maka data tersebut harus dikonversi menjadi representasi stringnya. Berikut ini adalah fungsi-fungsi konversi yang dapat kita gunakan:

```

// Konversi string s menjadi bentuk numerik integer-nya
int atoi(const char *s);

// Konversi string s menjadi bentuk numerik double-nya
double atof(const char *s);

// Konversi numerik double menjadi representasinya dalam bentuk
// null terminated string
char *gcvt(double value, int ndec, char *buf);

```

Fungsi atoi digunakan untuk mengkonversi nilai numerik integer dalam bentuk representasi string menjadi bilangan numerik integer sebenarnya. Prototype fungsi atoi dideklarasikan pada file header stdlib.h. Fungsi atof digunakan untuk mengubah nilai numerik

float dan *double* dalam bentuk representasi string menjadi nilai numerik double yang sesungguhnya. Prototype fungsi *atof* dideklarasikan dalam file header *math.h*. Jika fungsi *atoi* dan *atof* digunakan untuk mengkonversi nilai numerik dalam representasi string menjadi nilai numerik sesungguhnya, maka fungsi *gcvt* digunakan untuk mengubah nilai numerik menjadi representasi stringya.

Untuk lebih memahami ketiga fungsi konversi yang telah dibahas sebelumnya, berikut ini adalah contoh program untuk mempraktekan ketiga fungsi tersebut. Simpan program berikut ini dengan nama project *contoh16.ideal* pada direktori yang sama dengan file *screen.cpp*. Pada saat pembuatan project, pilihlah mode Emulation atau Floating Point pada group box Math Support di jendela New Target. Hal ini dilakukan karena pada program *contoh16.cpp* akan dilakukan perhitungan bilangan numerik real.

contoh16.cpp:

```

001 #include <math.h>           /* Deklarasi prototype atof          */
002 #include <stdlib.h>          /* Deklarasi prototype gcvt          */
003 #include "screen.cpp"         /* Fungsi anggota writeString       */
004
005 #define KEY_INT      0x16    /* Nomor interupsi keyboard        */
006 #define KEY_BACKSPACE 0x08    /* Tombol Backspace                 */
007 #define KEY_RETURN   0x0d    /* Tombol Enter                     */
008 #define KEY_TAB      0x09    /* Tombol Tab                      */
009 #define KEY_SPACE    0x20    /* Tombol spasi                    */
010
011 #define NULL        0x00    /* ASCII 0                         */
012 #define TRUE        1
013
014 UCHAR *getString(Screen *scr, UCHAR *str, UCHAR max);
015
016 int main(void)
017 {
018     Screen *layar = new Screen();
019     double luas, keliling, jari2;
020     UCHAR *str, str2[16];
021
022     layar->setMode(0x03);
023     layar->setCursorPos(5, 10);
024     layar->writeString("Luas dan Keliling Lingkaran");
025     layar->setCursorPos(6, 10);
026     layar->writeString("-----");
027     layar->setCursorPos(8, 10);
028     layar->writeString("Panjang jari-jari =");
029     layar->setCursorPos(8, 30);
030
031     /* Input jari-jari ke dalam variabel str */
032     str = getString(layar, str, 15);
033
034     /* Ubah str menjadi numerik double lalu simpan di jari2 */
035     jari2 = atof(str);
036
037     /* Hitung luas dan keliling lingkaran */
038     luas = M_PI * pow(jari2, 2);
039     keliling = 2 * M_PI * jari2;
040
041     layar->setCursorPos(9, 10);
042     layar->writeString("Luas =");
043     layar->setCursorPos(9, 30);

```

```

040
041     /* Ubah menjadi string, maks. angka 10 digit */
042     gcvt(luas, 10, str2);
043
044     layar->writeString(str2);
045     layar->setCursorPos(10, 10);
046     layar->writeString("Keliling           =");
047     layar->setCursorPos(10, 30);
048
049     /* Ubah menjadi string, maks. angka 10 digit */
050     gcvt(keliling, 10, str2);
051
052     layar->writeString(str2);
053     layar->setCursorPos(12, 10);
054     layar->writeString("Tekan ENTER ...");
055
056     getString(layar, str2, 0);
057
058     delete layar; return EXIT_SUCCESS;
059 }
060
061 UCHAR *getString(Screen *scr, UCHAR *str, UCHAR max)
062 {
063     UCHAR key, i, x, y;
064
065     key = i = 0;
066
067     while (TRUE)
068     {
069         asm mov ah, 0x00;
070         asm int KEY_INT;
071         asm mov key, al;
072
073         if ((key == KEY_BACKSPACE) && (i > 0))
074         {
075             scr->getCursorPos(&y, &x);
076             scr->setCursorPos(y, --x);
077             scr->writeChar(KEY_SPACE);
078
079             *(str + i) = NULL; i--;
080         }
081
082         if ((key >= 32) && (key <= 126) && (i < max))
083         {
084             scr->getCursorPos(&y, &x);
085             scr->writeChar(key);
086             scr->setCursorPos(y, ++x);
087
088             *(str + i) = key; i++;
089         }
090
091         if ((key == KEY_TAB) && (i < max))
092         {
093             scr->getCursorPos(&y, &x);
094             scr->writeChar(KEY_SPACE);

```

```

095         *(str + i) = KEY_SPACE; i++;
096     }
097
098     if (key == KEY_RETURN)
099     {
100         *(str + i) = NULL;
101         break;
102     }
103
104     if (i == max) *(str + i) = NULL;
105 }
106
107     return str;
108 }
```

Perhatikanlah baris 31 pada program contoh16.cpp! Pada baris tersebut, nilai str yang sebelumnya dimasukkan menggunakan fungsi getString diubah menjadi nilai numerik yang sesungguhnya lalu disimpan dalam variabel jari2. Kemudian perhatikan baris 42. Pada baris 42 tertulis gcvt(luas, 10, str). Maksudnya adalah nilai dari variabel luas akan diubah menjadi representasi stringnya yang disimpan pada variabel str. Banyak karakter yang dapat ditampung adalah 10 karakter. Demikian juga yang dilakukan pada baris 50.

3.7. Membuat Class untuk Operasi pada Keyboard

Setelah memahami teknik-teknik memasukan data melalui keyboard menggunakan teknik inline assembly, pada sub bab ini akan dibuat pustaka class (*class library*) yang mengenkapsulasi semua fungsi operasi keyboard yang telah dibuat sebelumnya. Jadi, fungsi getKey, getSpecialKey, getString, getPwdString dan getKeyState akan menjadi fungsi anggota class Keyboard. Class Keyboard akan bertindak seperti class Screen pada file screen.cpp, yaitu menyediakan antarmuka untuk operasi input menggunakan keyboard. Pada class Keyboard akan ditambahkan dua fungsi anggota baru, yaitu fungsi hideCursor (untuk menyembunyikan kursor) dan fungsi showCursor (untuk menampilkan kursor). Karena class Keyboard yang akan dibuat menggunakan class Screen maka file kode program class Keyboard, keyboard.cpp, harus disimpan dalam direktori yang sama. Berikut ini adalah kode program class Keyboard.

keyboard.cpp:

```

001 /*
002     keyboard.cpp
003     Class library untuk operasi input menggunakan keyboard.
004     Hak Cipta Pebi Yudha K.
005     April 2009
006
007     Disusun sebagai contoh program
008     Modul Praktikum Pemrograman C++ Lanjutan
009     AMIK BSI
010 */
011
012 #define KEY_INT          0x16 /* Nomor interupsi keyboard */
013 #define KEY_BACKSPACE    0x08 /* Tombol Backspace */
014 #define KEY_RETURN        0x0d /* Tombol Enter */
015 #define KEY_TAB           0x09 /* Tombol Tab */
016 #define KEY_SPACE          0x20 /* Tombol spasi */
017
018 #define STATE_RSHIFT      0x01 /* Status Shift Kanan */
019 #define STATE_LSHIFT      0x02 /* Status Shift Kiri */
```

```

020 #define STATE_CTRL      0x04 /* Status CTRL          */
021 #define STATE_ALT       0x08 /* Status ALT           */
022 #define STATE_SCROLL    0x10 /* Status Scroll Lock   */
023 #define STATE_NUM        0x20 /* Status Num Lock      */
024 #define STATE_CAPS       0x40 /* Status Caps Lock     */
025 #define STATE_INS        0x80 /* Status INS           */
026
027 #define NULL            0     /* ASCII 0              */
028 #define TRUE             1
029
030 class Keyboard
031 {
032     private:
033         Screen *screen;
034
035     public:
036     Keyboard(Screen *scr);      /* Konstruktor default */
037     ~Keyboard(void);           /* Destruktor default */
038
039     UCHAR getKey(void);
040     UCHAR getSpecialKey(void);
041     UCHAR *getString(UCHAR *str, UCHAR max);
042     UCHAR *getPwdString(UCHAR *pwd, UCHAR max);
043     UCHAR getKeyState(UCHAR key);
044
045     void hideCursor(void);
046     void showCursor(void);
047 };
048
049 Keyboard::Keyboard(Screen *scr)
050 {
051     this->screen = scr;
052
053     return;
054 }
055
056 Keyboard::~Keyboard(void)
057 {
058     return;
059 }
060
061 UCHAR Keyboard::getKey(void)
062 {
063     UCHAR key;
064
065     asm mov ah, 0x00; /* Register AH = 0          */
066     asm int KEY_INT;  /* Lakukan interupsi        */
067     asm mov key, al; /* Salin nilai register AH ke key */
068
069     return key;
070 }
071
072 UCHAR Keyboard::getSpecialKey(void)
073 {
074     UCHAR key;
075
076     asm mov ah, 0x00; /* Register AH = 0          */

```

```

077     asm int KEY_INT;      /* Lakukan interupsi 16 heksadesimal */
078     asm mov key, ah;      /* Salin nilai di AH ke variabel key */
079
080     return key;
081 }
082
083 UCHAR *Keyboard::getString(UCHAR *str, UCHAR max)
084 {
084     UCHAR key = 0;
085     UCHAR i, x, y;
086
087     i = 0;
088
089     while (TRUE)
090     {
091         asm mov ah, 0x00;
092         asm int KEY_INT;
093         asm mov key, al;
094
095         if ((key == KEY_BACKSPACE) && (i > 0))
096         {
097             this->screen->getCursorPos(&y, &x);
098             this->screen->setCursorPos(y, --x);
099             this->screen->writeChar(KEY_SPACE);
100
101             *(str + i) = NULL; i--;
102         }
103
104         if ((key >= 32) && (key <= 126) && (i < max))
105         {
106             this->screen->getCursorPos(&y, &x);
107             this->screen->writeChar(key);
108             this->screen->setCursorPos(y, ++x);
109
110             *(str + i) = key; i++;
111         }
112
113         if ((key == KEY_TAB) && (i < max))
114         {
115             this->screen->getCursorPos(&y, &x);
116             this->screen->writeChar(KEY_SPACE);
117             this->screen->setCursorPos(y, ++x);
118
119             *(str + i) = KEY_SPACE; i++;
120         }
121
122         if (key == KEY_RETURN)
123         {
124             *(str + i) = NULL;
125             break;
126         }
127
128         if (i == max) *(str + i) = NULL;
129     }
130
131     return str;
132 }

```

```

133
134 UCHAR *Keyboard::getPwdString(UCHAR *pwd, UCHAR max)
135 {
136     UCHAR key, i, x, y;
137
138     key = i = 0;
139
140     while (TRUE)
141     {
142         asm mov ah, 0x00;
143         asm int KEY_INT;
144         asm mov key, al;
145
146         if ((key == KEY_BACKSPACE) && (i > 0))
147         {
148             this->screen->getCursorPos(&y, &x);
149             this->screen->setCursorPos(y, --x);
150             this->screen->writeChar(KEY_SPACE);
151
152             *(pwd + i) = NULL; i--;
153         }
154
155         if ((key >= 32) && (key <= 126) && (i < max))
156         {
157             this->screen->getCursorPos(&y, &x);
158             this->screen->writeChar('*');
159             this->screen->setCursorPos(y, ++x);
160
161             *(pwd + i) = key; i++;
162         }
163
164         if ((key == KEY_TAB) && (i < max))
165         {
166             this->screen->getCursorPos(&y, &x);
167             this->screen->writeChar('*');
168             this->screen->setCursorPos(y, ++x);
169
170             *(pwd + i) = '*'; i++;
171         }
172
173         if (key == KEY_RETURN)
174         {
175             *(pwd + i) = NULL;
176             break;
177         }
178
179         if (i == max) *(pwd + i) = NULL;
180     }
181
182     return pwd;
183 }
184
185 UCHAR Keyboard::getKeyState(UCHAR key)
186 {
187     UCHAR state;
188
189     asm mov ah, 0x02; /* Register AH = 2 */
```

```

190     asm int KEY_INT;      /* Lakukan interupsi 16 heksadesimal */
191     asm and al, key;    /* AND-kan register AL sesuai tombol */
192     asm mov state, al;  /* Salin nilai register AL ke state */
193
194     return state;
195 }
196
197 void Keyboard::hideCursor(void)
198 {
199     asm mov ah, 0x01;
200     asm mov ch, 0x20;
201     asm mov cl, 0x00;
202     asm int VIDEO_INT;
203
204     return;
205 }
206
207 void Keyboard::showCursor(void)
208 {
209     asm mov ah, 0x01;
210     asm mov ch, 0x06;
211     asm mov cl, 0x07;
212     asm int VIDEO_INT;
213
214     return;
215 }

```

Untuk mempraktekkan cara menggunakan class Keyboard, berikut ini akan diberikan contoh program. Program berikut ini adalah program sederhana untuk menghitung besar beda potensial dengan mengalikan besar tahanan dengan besar arus listrik. Pengguna hanya harus memasukan nilai kuat arus listrik dan besar tahanan. Simpan project contoh17.ide dan kode program contoh17.cpp berikut ini pada direktori yang sama dengan file screen.cpp dan keyboard.cpp.

contoh17.cpp:

```

01 #include <math.h>
02 #include <stdlib.h>
03 #include "screen.cpp"
04 #include "keyboard.cpp"
05
06 int main(void)
07 {
08     Screen *layar = new Screen();
09     Keyboard *tombol = new Keyboard(layar);
10
11     unsigned long int v, i, r;
12     UCHAR *str, str2[16];
13
14     layar->setMode(0x03);
15     layar->setCursorPos(5, 14);
16     layar->writeString("Menghitung Beda Potensial");
17     layar->setCursorPos(7, 14);
18     layar->writeString("Besar arus (ampere)    =");
19     layar->setCursorPos(7, 38);
20
21     str = tombol->getString(str2, 10);

```

```

22     i = atoi(str);
23
24     layar->setCursorPos(8, 14);
25     layar->writeString("Besar tahanan (ohm)    =");
26     layar->setCursorPos(8, 38);
27
28     str = tombol->getString(str2, 10);
29     r = atoi(str);
30     v = i * r;
31
32     gcvt(v, 10, str2);
33
34     layar->setCursorPos(9, 14);
35     layar->writeString("Beda potensial (volt)   =");
36     layar->setCursorPos(9, 38);
37     layar->writeString(str2);
38     layar->setCursorPos(11, 14);
39     layar->writeString("Tekan SHIFT ...");
40     tombol->hideCursor();
41
42     while (TRUE)
43     {
44         if ((tombol->getKeyState(STATE_RSHIFT)) ||
45             (tombol->getKeyState(STATE_LSHIFT)))
46             break;
47     }
48
49     delete layar; delete tombol;
50     return EXIT_SUCCESS;
51 }
```

3.8. Latihan-latihan Bab III

1. Jelaskan prosedur untuk menjalankan interupsi 16 heksadesimal servis 0!
2. Jelaskan interupsi yang digunakan untuk mengetahui status tombol Caps Lock dan Num Lock dalam keadaan ON atau OFF menggunakan diagram alur (*flowchart*)!
3. Perhatikan kode program contoh12.cpp! Ubah fungsi getKey yang ditulis dengan teknik inline assembly menjadi menggunakan fungsi standar int16!
4. Perhatikan kode program contoh15.cpp! Ubah fungsi getKeyState yang ditulis dengan teknik inline assembly menjadi menggunakan fungsi standar int16!
5. Jelaskan algoritma fungsi getString yang digunakan untuk memasukan string menggunakan diagram alur (*flowchart*)!
6. Jelaskan prosedur yang harus dilakukan untuk memasukan data numerik menggunakan fungsi getString menggunakan diagram alur (*flowchart*)!
7. Buatlah program dengan rancangan input dan output seperti dibawah ini menggunakan class Screen dan class Keyboard!

Rancangan input:

Siapakah orang yang menciptakan bahasa C++?
 [A] Dennis Ritchie
 [B] Kenneth Thompson
 [C] Richard Stallman
 [D] Bjarne Stroustrup

Jawaban Anda: [...] <input dalam kotak hanya satu karakter>

Jika jawaban salah muncul pesan:

Salah, jawaban yang benar adalah D !

Jika jawaban benar muncul pesan:

Selamat, Anda benar!

Jika jawaban selain A, B, C, atau D maka muncul pesan

Pilihlah A, B, C, atau D! Ulangi (Y/T)? [...]

Jika pengguna memilih Y atau y maka layar dibersihkan dan program akan diulang. Jika memilih selain Y atau y maka program berhenti.

8. Buatlah program dengan rancangan input dan output seperti dibawah ini menggunakan class Screen dan Keyboard!

Rancangan input:

Username	:	Jully Triansyah	<input string>
Password	:	*****	<input string password>

Rancangan output:

Selamat datang JULLY TRIANSYAH <username sesuai input>
Tekan F1 untuk logout atau F2 untuk menghentikan program.

Jika F1 ditekan maka layar dibersihkan dan program diulang. Jika F2 ditekan maka program berhenti.

9. Buatlah program untuk menghitung resistansi resistor dengan tampilan berikut ini

Besar voltase (volt)	:	12.5	<input>
Besar arus (ampere)	:	3.5	<input>
Resistansi	:	3.571 Ω	<output>

10. Ubahlah fungsi getPwdString pada kode program contoh14.cpp yang sebelumnya menggunakan teknik inline assembly menjadi menggunakan fungsi standar int86!

BAB IV

Mendeteksi dan Menggunakan Mouse

4.1. Interupsi DOS untuk Mendeteksi, Mengaktifkan dan Menonaktifkan Mouse

Disk Operating System (DOS) baik pada mode real maupun mode terproteksi (DPMI) menyediakan sebuah *software interrupt* khusus untuk mengaktifkan dan menonaktifkan mouse (tetikus), yaitu interupsi 33 heksadesimal. Interupsi 33 heksadesimal menyediakan servis untuk mendeteksi mouse, menampilkan atau menyembunyikan posisi kursor, mengetahui posisi kursor mouse, mengetahui status penekanan tombol mouse, mengetahui jenis mouse, mengaktifkan dan menonaktifkan driver mouse, membatasi koordinat mouse dan lain-lain.

Karena pada awalnya DOS tidak mendukung penggunaan mouse secara *built-in*, maka dibutuhkan *driver* khusus agar mouse bisa diprogram menggunakan interupsi 33 heksadesimal. Driver mouse harus *di-install* terlebih dahulu jika kita ingin memprogram mouse pada mode real DOS. Caranya adalah dengan menambahkan perintah DEVICE=MOUSE.SYS pada file CONFIG.SYS jika driver yang dimiliki berbentuk file MOUSE.SYS, atau dengan menjalankan perintah MOUSE pada *command prompt* jika driver yang dimiliki adalah MOUSE.COM. Namun jika kita menjalankan DOS pada mode DPMI Microsoft Windows, kita tidak membutuhkan driver mouse karena Microsoft Windows telah menyediakan dan menjalankan driver mouse tersebut pada saat *start-up*.

4.2. Mendeteksi Mouse dan Jumlah Tombol pada Mouse

Untuk mengetahui apakah mouse pada PC siap digunakan atau sudah terinstal dengan benar, kita dapat menggunakan interupsi 33 heksadesimal servis 0. Interupsi 33 heksadesimal servis 0 dapat digunakan untuk mendeteksi mouse dan mengetahui jumlah tombol yang dimiliki oleh mouse tersebut. Berikut ini akan dijelaskan prosedur untuk menjalankan interupsi 33 heksadesimal servis 0.

- ✓ Register AX harus bernilai 0.
- ✓ Jalankan interupsi 33 heksadesimal.

Setelah interupsi dijalankan:

- ✓ Register AX akan bernilai 0 jika mouse atau driver-nya tidak terinstal.
- ✓ Register AX akan bernilai 0xffff heksadesimal jika mouse dan driver-nya terinstal dengan benar.
- ✓ Register BX berisi nilai yang menunjukkan jumlah tombol mouse.
- ✓ Pointer mouse akan direset dan disembunyikan. Koordinat awal pointer mouse adalah ditengah layar.

Berikut ini adalah contoh program untuk mempraktekan interupsi 33 heksadesimal servis 0. Simpan project berikut ini dengan nama contoh18.ide pada direktori yang sama dengan file screen.cpp dan keyboard.cpp.

contoh18.cpp:

```
01 #include <stdlib.h>
02 #include "screen.cpp"
03 #include "keyboard.cpp"
04
05 #define MOUSE_INT 0x33 /* Nomor interupsi mouse */
06
```

```

07 UCHAR detectMouse(UCHAR *btn);
08
09 int main(void)
10 {
11     Screen *layar = new Screen();
12     Keyboard *tombol = new Keyboard(layar);
13     UCHAR str[5];
14     UCHAR status, button;
15
16     /* Panggil fungsi deteksi mouse */
17     status = detectMouse(&button);
18
19     layar->setMode(0x03);
20     layar->setCursorPos(5, 14);
21     layar->writeString("Deteksi Mouse");
22     layar->setCursorPos(7, 14);
23     layar->writeString("Status:");
24     layar->setCursorPos(8, 14);
25     layar->writeString("Tombol:");
26     layar->setCursorPos(7, 22);
27
28     if (status)
29         layar->writeString("Siap digunakan");
30     else
31         layar->writeString("Tidak terinstal");
32
33     layar->setCursorPos(8, 22);
34     layar->writeChar(button | 0x30);
35     tombol->getString(str, 0);
36
37     delete layar;
38     delete tombol;
39     return EXIT_SUCCESS;
40 }
41
42 UCHAR detectMouse(UCHAR *btn)
43 {
44     UCHAR state, button;
45
46     asm mov ah, 0x00;      /* Register AH = 0          */
47     asm mov al, 0x00;      /* Register AL = 0, AH:AL = 0:0 */
48     asm int MOUSE_INT;    /* Laksanakan interupsi 0x33 */
49     asm mov state, al;    /* Salin nilai AL ke state */
50     asm mov button, bl;   /* Salin nilai BL ke button */
51
52     *btn = button;
53     return state;
54 }
```

Fungsi `detectMouse` pada contoh18.cpp menerima sebuah parameter yang dikirimkan secara referensi, yaitu `btn`. Setelah fungsi `detectMouse` dijalankan maka parameter `btn` berisi nilai yang menunjukkan jumlah tombol mouse. Fungsi `detectMouse` akan mengembalikan nilai lebih dari 0 jika mouse dan drivernya terinstal dengan benar dan akan mengembalikan nilai 0 jika mouse atau drivernya tidak terinstal.

4.3. Menampilkan dan Menyembunyikan Pointer Mouse

Pointer mouse sama halnya dengan kursor keyboard yang dapat disembunyikan dan ditampilkan kembali. Untuk menampilkan pointer mouse digunakan interupsi 33 heksadesimal servis 1, sedangkan untuk menyembunyikan pointer mouse digunakan interupsi 33 heksadesimal servis 2. Berikut ini adalah prosedur untuk menampilkan dan menyembunyikan mouse.

Menampilkan pointer mouse:

- ↪ Register AH harus berisi nilai 0.
- ↪ Register AL harus berisi nilai 1.
- ↪ Lakukan interupsi 33 heksadesimal.

Menyembunyikan pointer mouse:

- ↪ Register AH harus berisi nilai 0.
- ↪ Register AL harus bernilai 2.
- ↪ Lakukan interupsi 33 heksadesimal.

Berikut ini adalah contoh program untuk mempraktekkan cara menampilkan pointer mouse. Simpan project berikut ini dengan nama contoh19.ide pada direktori yang sama dengan screen.cpp.

contoh19.cpp:

```
01 #include <dos.h>
02 #include <stdlib.h>
03 #include "screen.cpp"
04
05 #define MOUSE_INT 0x33
06
07 UCHAR detectMouse(UCHAR *btn);
08 void showMouse(void);
09
10 int main (void)
11 {
12     Screen *layar = new Screen();
13     UCHAR state, btn;
14
15     layar->setMode(0x03);
16     layar->setCursorPos(4, 14);
17
18     state = detectMouse(&btn);
19
20     if (state)
21     {
22         layar->writeString("Mouse siap digunakan.");
23         showMouse();
24     }
25     else
26     {
27         layar->writeString("Mouse tidak siap.");
28     }
29
30     delay(5000);
31     delete layar; return EXIT_SUCCESS;
32 }
33
34 UCHAR detectMouse(UCHAR *btn)
35 {
```

```

36    UCHAR state, button;
37
38    asm mov ah, 0x00;      /* Register AH = 0          */
39    asm mov al, 0x00;      /* Register AL = 0, AH:AL = 0:0 */
40    asm int MOUSE_INT;   /* Lakukan interupsi 0x33    */
41    asm mov state, al;   /* Salin nilai AL ke state */
42    asm mov button, bl;  /* Salin nilai BL ke button */
43
44    *btn = button;
45    return state;
46 }
47
48 void showMouse(void)
49 {
50    asm mov ah, 0x00;      /* Register AH = 0          */
51    asm mov al, 0x01;      /* Register AL = 0, AX = 0   */
52    asm int MOUSE_INT;   /* Lakukan interupsi        */
53
54    return;
55 }
```

Program contoh19.cpp ketika dijalankan akan mendeteksi mouse. Jika mouse siap digunakan maka pointer mouse akan dimunculkan menggunakan fungsi showMouse, tetapi jika mouse tidak terdeteksi maka akan ditampilkan pesan bahwa mouse tidak bisa digunakan. Agar pointer mouse dapat ditampilkan, program contoh19.cpp harus dijalankan dalam mode *full screen*, caranya adalah dengan menekan tombol kombinasi ALT+Enter. Program akan berhenti secara otomatis setelah lima detik. Setelah bisa menampilkan pointer mouse, program contoh20.ide berikut ini akan menunjukkan cara menyembunyikan pointer mouse.

contoh20.cpp:

```

01 #include <dos.h>
02 #include <stdlib.h>
03 #include "screen.cpp"
04
05 #define MOUSE_INT 0x33
06
07 UCHAR detectMouse(UCHAR *btn);
08 void showMouse(void);
09 void hideMouse(void);
10
11 int main (void)
12 {
13     Screen *layar = new Screen();
14     UCHAR state, btn;
15
16     layar->setMode(0x03);
17     layar->setCursorPos(4, 14);
18
19     state = detectMouse(&btn);
20
21     if (state)
22     {
23         layar->writeString("Mouse siap digunakan.");
24         layar->setCursorPos(6, 14);
25         layar->writeString("Pointer mouse ditampilkan.");
```

```

26     showMouse(); delay(7000);
27
28     layar->setCursorPos(6, 14);
29     layar->writeString("Pointer mouse disembunyikan.");
30     hideMouse();
31 }
32 }
33 else
34 {
35     layar->writeString("Mouse tidak siap.");
36 }
37
38 delay(5000);
39 delete layar; return EXIT_SUCCESS;
40 }
41
42 UCHAR detectMouse(UCHAR *btn)
43 {
44     UCHAR state, button;
45
46     asm mov ah, 0x00;      /* Register AH = 0          */
47     asm mov al, 0x00;      /* Register AL = 0, AH:AL = 0:0 */
48     asm int MOUSE_INT;    /* Laksanakan interupsi 0x33   */
49     asm mov state, al;    /* Salin nilai AL ke state   */
50     asm mov button, bl;   /* Salin nilai BL ke button   */
51
52     *btn = button;
53     return state;
54 }
55
56 void showMouse(void)
57 {
58     asm mov ah, 0x00;      /* Register AH = 0          */
59     asm mov al, 0x01;      /* Register AL = 1, AH:AL = AX = 1 */
60     asm int MOUSE_INT;    /* Tampilkan mouse           */
61
62     return;
63 }
64
65 void hideMouse(void)
66 {
67     asm mov ah, 0x00;      /* Register AH = 0          */
68     asm mov al, 0x02;      /* Register AL = 2, AH:AL = AX = 2 */
69     asm int MOUSE_INT;    /* Sembunyikan mouse         */
70
71     return;
72 }

```

Program contoh20.cpp menambahkan satu fungsi baru, yaitu fungsi hideMouse untuk menyembunyikan pointer mouse. Alur program contoh20.cpp adalah sebagai berikut:

1. Deteksi mouse, jika mouse tidak siap digunakan maka pesan bahwa mouse tidak siap digunakan ditampilkan. Kemudian program berhenti secara otomatis setelah jeda lima detik.
2. Jika mouse terdeteksi, maka pointer mouse akan ditampilkan. Setelah jeda selama tujuh detik pointer mouse akan disembunyikan. Setelah itu program berhenti secara otomatis

setelah jeda lima detik.

4.4. Mengaktifkan dan Menonaktifkan Driver Mouse

Mouse dan driver-nya yang telah terinstal dengan benar dapat diaktifkan dan dinonaktifkan. Jika suatu driver mouse dinonaktifkan maka pointernya tidak bisa digerakan dan koordinatnya tidak akan berubah, walaupun mouse digerak-gerakan. Untuk mengaktifkan driver mouse digunakan interupsi 33 heksadesimal servis 20 heksadesimal. Sedangkan untuk menonaktifkan driver mouse dapat digunakan interupsi 33 heksadesimal servis 1f heksadesimal. Berikut ini adalah prosedur untuk mengaktifkan dan menonaktifkan mouse:

Mengaktifkan driver mouse:

- ↪ Register AH harus bernilai 0.
- ↪ Register AL harus bernilai 20 heksadesimal.
- ↪ Laksanakan interupsi 33 heksadesimal.

Menonaktifkan driver mouse:

- ↪ Register AH harus bernilai 0.
- ↪ Register AL harus bernilai 1f heksadesimal.
- ↪ Laksanakan interupsi 33 heksadesimal.

Setelah interupsi 33 heksadesimal servis 1f heksadesimal dijalankan:

- ↪ Jika interupsi berhasil dilaksanakan maka register AX bernilai 001f heksadesimal.
- ↪ Jika interupsi gagal dilaksanakan maka register AX bernilai ffff heksadesimal.

Untuk mempraktekkan prosedur mengaktifkan dan menonaktifkan mouse, pelajari kode program project contoh21.ide berikut ini. Simpan kode programnya pada direktori yang sama dengan file screen.cpp.

contoh21.cpp:

```
01 #include <dos.h>
02 #include <stdlib.h>
03 #include "screen.cpp"
04
05 #define MOUSE_INT 0x33
06
07 UCHAR detectMouse(UCHAR *btn);
08 void showMouse(void);
09 void enableMouse(void);
10 UCHAR disableMouse(void);
11
12 int main (void)
13 {
14     Screen *layar = new Screen();
15     UCHAR state, btn;
16
17     layar->setMode(0x03);
18     layar->setCursorPos(4, 14);
19
20     state = detectMouse(&btn);
21
22     if (state)
23     {
24         layar->writeString("Mouse siap digunakan.");
25         layar->setCursorPos(6, 14);
```

```

26     layar->writeString("Driver mouse dinonaktifkan.");
27
28     showMouse(); disableMouse();
29     delay(7000);
30
31     layar->setCursorPos(6, 14);
32     layar->writeString("Driver mouse diaktifkan kembali.");
33
34     enableMouse();
35 }
36 else
37 {
38     layar->writeString("Mouse tidak siap.");
39 }
40
41 delay(5000);
42 delete layar; return EXIT_SUCCESS;
43 }
44
45 UCHAR detectMouse(UCHAR *btn)
46 {
47     UCHAR state, button;
48
49     asm mov ah, 0x00; /* Register AH = 0 */
50     asm mov al, 0x00; /* Register AL = 0, AH:AL = 0:0 */
51     asm int MOUSE_INT; /* Lakukan interupsi 0x33 */
52     asm mov state, al; /* Salin nilai AL ke state */
53     asm mov button, bl; /* Salin nilai BL ke button */
54
55     *btn = button;
56     return state;
57 }
58
59 void showMouse(void)
60 {
61     asm mov ah, 0x00; /* Register AH = 0 */
62     asm mov al, 0x01; /* Register AL = 1, AH:AL = AX = 1 */
63     asm int MOUSE_INT; /* Tampilkan pointer mouse */
64
65     return;
66 }
67
68 void enableMouse(void)
69 {
70     asm mov ah, 0x00; /* Register AH = 0 */
71     asm mov al, 0x20; /* Register AL = 0x20, AX = 0x0020 */
72     asm int MOUSE_INT; /* Aktifkan driver mouse */
73
74     return;
75 }
76
77 UCHAR disableMouse(void)
78 {
79     UCHAR state;
80
81     asm mov ah, 0x00; /* Register AH = 0 */
82     asm mov al, 0x1f; /* Register AL = 0x1f, AX = 0x001f */

```

```

83     asm int MOUSE_INT;      /* Non-aktifkan driver mouse      */
84     asm mov state, al;      /* Salin isi AL ke state          */
85
86     return state;
87 }

```

Pada program contoh21.cpp diatas, fungsi enableMouse digunakan untuk mengaktifkan driver mouse dan fungsi disableMouse digunakan untuk menonaktifkan driver mouse. Perbedaan fungsi enableMouse dengan fungsi disableMouse yang telah dibuat adalah fungsi disableMouse mengembalikan nilai unsigned character yang menunjukkan apakah interupsi berhasil dilakukan atau gagal, sedangkan fungsi enableMouse tidak mengembalikan nilai apapun.

4.5. Mengetahui Koordinat Pointer Mouse

Seperti halnya kursor keyboard yang ditampilkan di layar monitor, pointer mouse pun menempati koordinat tertentu di layar. Untuk mengetahui letak atau posisi koordinat pointer mouse dilayar dapat digunakan interupsi 33 heksadesimal servis 3. Berikut ini adalah prosedur untuk mengetahui koordinat pointer mouse.

- ❑ Register AH harus bernilai 0.
- ❑ Register AL harus bernilai 3 heksadesimal.
- ❑ Lakukan interupsi 33 heksadesimal.

Setelah interupsi dilaksanakan:

- ❑ Register CX menunjukkan posisi horizontal pointer mouse (sumbu X).
- ❑ Register DX menunjukkan posisi vertikal pointer mouse (sumbu Y).
- ❑ Register BX menunjukkan tombol mouse yang ditekan. Jika nilai register BX sama dengan 1 berarti tombol kiri mouse yang ditekan. Jika tombol kanan mouse yang ditekan, nilai register BX sama dengan 2.
- ❑ Posisi vertikal dan horizontal pointer mouse dimulai dari posisi 0.

Dari prosedur yang telah dijelaskan diatas, dapat disimpulkan bahwa interupsi 33 heksadesimal servis 3 selain dapat digunakan untuk mengetahui posisi koordinat pointer mouse, juga dapat digunakan untuk mengetahui status penekanan tombol mouse. Berikut ini adalah contoh program untuk mempraktekan interupsi 33 heksadesimal servis 3. Simpan project contoh22.ide berikut ini pada direktori yang sama dengan file screen.cpp dan keyboard.cpp serta pilihlah Floating Point atau Emulation di group box Math Support pada jendela New Target.

contoh22.cpp:

```

001 #include <stdlib.h>
002 #include "keyboard.cpp"
003 #include "screen.cpp"
004
005 #define MOUSE_INT          0x33
005 #define MOUSE_RIGHT_CLICK 0x02
006
007 #define USHORT unsigned short int
008
009 UCHAR detectMouse(UCHAR *btn);
010 void getMousePos(USHORT *row, USHORT *col, UCHAR *btn);
011 void showMouse(void);
012
013 int main(void)
014 {

```

```

015 Screen *layar = new Screen();
016 Keyboard *tuts = new Keyboard(layar);
017
018 USHORT baris, kolom;
019 UCHAR status, button, str[5];
020 status = detectMouse(&button);
021
022 layar->setMode(0x03);
023 layar->setCursorPos(4, 14);
024 tuts->hideCursor();
025
026 if (!status)
027 {
028     layar->writeString("Mouse tidak siap! Tekan ENTER ...");
029     tuts->getString(str, 0);
030
031     delete tuts; delete layar;
032     exit(EXIT_FAILURE);
033 }
034
035 layar->writeString("Deteksi Koordinat Pointer Mouse");
036 layar->setCursorPos(6, 14);
037 layar->writeString("X:");
038 layar->setCursorPos(7, 14);
039 layar->writeString("Y:");
040 layar->setCursorPos(9, 14);
041 layar->writeString("Klik kanan untuk berhenti");
042 showMouse();
043
044 while (TRUE)
045 {
046     /* Periksa koordinat dan penekanan tombol mouse */
047     getMousePos(&baris, &kolom, &button);
048
049     baris /= 8; kolom /= 8; /* kolom dan baris dibagi 8 */
050
051     gcvt((double) kolom, 2, str);
052     layar->setCursorPos(6, 17); layar->writeString(" ");
053     layar->setCursorPos(6, 17); layar->writeString(str);
054
055     gcvt((double) baris, 2, str);
056     layar->setCursorPos(7, 17); layar->writeString(" ");
057     layar->setCursorPos(7, 17); layar->writeString(str);
058
059     if (button == MOUSE_RIGHT_CLICK) break;
060 }
061
062 delete tuts; delete layar;
063 return EXIT_SUCCESS;
064
065 }
066
067 UCHAR detectMouse(UCHAR *btn)
068 {
069     UCHAR state, button;
070     asm mov ah, 0x00;    /* Register AH = 0           */
071     asm mov al, 0x00;    /* Register AL = 0, AH:AL = 0:0 */
072     asm int MOUSE_INT; /* Laksanakan interupsi 0x33   */

```

```

074     asm mov state, al;    /* Salin nilai AL ke state      */
075     asm mov button, bl;  /* Salin nilai BL ke button    */
076
077     *btn = button;
078     return state;
079 }
080
081 void getMousePos(USHORT *row, USHORT *col, UCHAR *btn)
082 {
083     USHORT x, y;
084     UCHAR button;
085
086     asm mov ah, 0x00;      /* Register AH = 0           */
087     asm mov al, 0x03;      /* Register AL = 0, AH:AL = AX = 0   */
088     asm int MOUSE_INT;    /* Lakukan interupsi 0x33       */
089     asm mov x, cx;        /* Salin CX ke x, posisi horizontal */
090     asm mov y, dx;        /* Salin DX ke y, posisi vertikal   */
091     asm mov button, bl;   /* Status penekanan tombol      */
092
093     *row = y; *col = x; *btn = button;
094
095     return;
096 }
097
098 void showMouse(void)
099 {
100     asm mov ah, 0x00;
101     asm mov al, 0x01;
102     asm int MOUSE_INT;
103
104     return;
105 }
```

Pada program contoh22.cpp, fungsi getMousePos menerima tiga parameter, yaitu row, col dan btn yang dikirimkan secara referensi. Parameter row dan col yang bertipe unsigned integer digunakan untuk menyimpan posisi baris dan kolom pointer mouse, sedangkan parameter btn yang bertipe unsigned character digunakan untuk mengetahui apakah tombol kiri atau kanan ditekan. Perhatikanlah baris 51! Pada baris 51, variabel baris dan kolom dibagi dengan 8 setelah fungsi getMousePos dijalankan karena nilai baris yang dikembalikan akan berkisar dari 0 s.d 479 dan nilai kolom yang dikembalikan berkisar antara 0 s.d 639. Berikut ini adalah alur program contoh22.cpp:

1. Deteksi mouse. Jika mouse tidak siap digunakan maka program dihentikan.
2. Jika mouse siap digunakan maka lakukan perulangan terus-menerus untuk memeriksa koordinat pointer mouse.
3. Tampilkan posisi baris dan kolom pointer mouse.
4. Periksa apakah tombol kanan mouse diklik. Jika tombol kanan mouse diklik maka hentikan perulangan dan program selesai.

4.6. Memindahkan Koordinat Pointer Mouse

Koordinat pointer mouse dapat dipindahkan tanpa harus menggerak-gerakan mouse secara manual. Untuk memindahkan posisi koordinat pointer mouse dapat digunakan interupsi 33 heksadesimal servis 4. Berikut ini adalah prosedur yang harus dilakukan untuk memindahkan pointer menggunakan interupsi 33 heksadesimal servis 4.

- ☒ Register AH harus bernilai 0.
- ☒ Register AL harus bernilai 4 heksadesimal.
- ☒ Register CX menentukan posisi baris pointer mouse.
- ☒ Register DX menentukan posisi kolom pointer mouse.
- ☒ Lakukan interupsi 33 heksadesimal.

Berikut ini adalah contoh program untuk mempraktekan interupsi 33 heksadesimal servis 4. Simpan project contoh23.ide berikut ini pada direktori yang sama dengan file screen.cpp dan keyboard.cpp.

contoh23.cpp:

```

001 #include <stdlib.h>
002 #include <time.h> /* Prototype fungsi randomize dan rand */
003 #include "screen.cpp"
004 #include "keyboard.cpp"
005
006 #define MOUSE_INT          0x33
007 #define MOUSE_LEFT_CLICK   0x01
008
009 #define USHORT unsigned short int
010
011 UCHAR detectMouse(UCHAR *btn);
012 void getMousePos(USHORT *row, USHORT *col, UCHAR *btn);
013 void setMousePos(USHORT row, USHORT col);
014 void showMouse(void);
015
016 int main(void)
017 {
018     Screen *layar = new Screen();
019     Keyboard *tombol = new Keyboard(layar);
020
021     USHORT baris, kolom;
022     UCHAR status, button, str[5];
023
024     status = detectMouse(&button);
025
026     layar->setMode(0x03);
027     layar->setCursorPos(4, 14);
028     tombol->hideCursor();
029
030     if (!status)
031     {
032         layar->writeString("Mouse tidak siap! Tekan ENTER ...");
033         tombol->getString(str, 0);
034
035         delete tombol; delete layar;
036         exit(EXIT_FAILURE);
037     }
038
039     layar->writeString("Klik kiri untuk berhenti.");
040     showMouse(); /* Tampilkan pointer mouse */
041     randomize(); /* Inisialisasi angka random */
042
043     while (TRUE)
044     {
045         baris = rand() % 480;

```

```

046     kolom = rand() % 640;
047
048     setMousePos(baris, kolom);
049     getMousePos(&baris, &kolom, &button);
050
051     if (button == MOUSE_LEFT_CLICK) break;
052 }
053
054 delete tombol; delete layar;
055 return EXIT_SUCCESS;
056 }
057
058 UCHAR detectMouse(UCHAR *btn)
059 {
060     UCHAR state, button;
061
062     asm mov ah, 0x00; /* Register AH = 0 */
063     asm mov al, 0x00; /* Register AL = 0, AH:AL = 0:0 */
064     asm int MOUSE_INT; /* Lakukan interupsi 0x33 */
065     asm mov state, al; /* Salin nilai AL ke state */
066     asm mov button, bl; /* Salin nilai BL ke button */
067
068     *btn = button;
069     return state;
070 }
071
072 void getMousePos(USHORT *row, USHORT *col, UCHAR *btn)
073 {
074     USHORT x, y;
075     UCHAR button;
076
077     asm mov ah, 0x00; /* Register AH = 0 */
078     asm mov al, 0x03; /* Register AL = 3, AH:AL = AX = 3 */
079     asm int MOUSE_INT; /* Lakukan interupsi 0x33 */
080     asm mov x, cx; /* Salin nilai register CX ke x */
081     asm mov y, dx; /* Salin nilai register DX ke y */
082     asm mov button, bl; /* Salin nilai reg. DL ke button */
083
084     *row = y; *col = x;
085     *btn = button;
086
087     return;
088 }
089
090 void setMousePos(USHORT row, USHORT col)
091 {
092     asm mov ax, 0x0004; /* Register AX = 4 */
093     asm mov cx, col; /* Salin nilai col ke register CX */
094     asm mov dx, row; /* Salin nilai row ke register DX */
095     asm int MOUSE_INT; /* Pindahkan posisi pointer mouse */
096
097     return;
098 }
099
100 void showMouse(void)
101 {
102     asm mov ah, 0x00; /* Register AH = 0 */

```

```

103     asm mov al, 0x01;      /* Register AL = 1, AH:AL = AX = 1 */
104     asm int MOUSE_INT;    /* Tampilkan pointer mouse           */
105
106     return;
107 }

```

Fungsi setMousePos pada program contoh23.cpp diatas digunakan untuk memindahkan pointer mouse. Fungsi setMousePos menerima parameter row dan col yang bertipe unsigned integer yang dikirimkan secara nilai. Parameter row digunakan untuk menentukan posisi vertikal pointer mouse, sedangkan parameter col digunakan untuk menentukan posisi horizontal mouse. Program contoh23.cpp baru berhenti ketika pengguna menekan tombol kiri mouse.

4.7. Membatasi Posisi Horizontal dan Vertikal Mouse

Letak koordinat pointer mouse dapat dibatasi sehingga posisi vertikal dan horizontal pointer mouse tidak bisa melewati area tertentu pada layar yang sudah dibatasi. Untuk membatasi posisi horizontal pointer mouse dapat digunakan interupsi 33 heksadesimal servis 7, sedangkan untuk membatasi posisi vertikal pointer mouse digunakan interupsi 33 heksadesimal servis 8. Berikut ini adalah prosedur untuk membatasi koordinat pointer mouse.

Membatasi posisi horizontal pointer mouse:

- Register AH harus bernilai 0.
- Register AL harus bernilai 7 heksadesimal.
- Register CX menentukan batas minimal posisi horizontal pointer mouse.
- Register DX menentukan batas maksimal posisi horizontal pointer mouse.

Membatasi posisi vertikal pointer mouse:

- Register AH harus bernilai 0.
- Register AL harus bernilai 8 heksadesimal.
- Register CX menentukan batas minimal posisi vertikal pointer mouse.
- Register DX menentukan batas maksimal posisi vertikal pointer mouse.

Catatan: Jika batas minimal lebih besar daripada batas maksimal, maka akan ditukar.

Untuk mempraktekkan teori yang telah dijelaskan diatas, project contoh24.ide berikut ini memberikan contoh membatasi posisi vertikal pointer mouse antara baris 10 sampai 15 dan posisi horizontal antara kolom 30 sampai 60. Simpan project contoh24.ide berikut ini dalam direktori yang sama dengan file screen.cpp dan keyboard.cpp.

contoh24.cpp:

```

001 #include <stdlib.h>
002 #include "screen.cpp"
003 #include "keyboard.cpp"
004
005 #define MOUSE_INT          0x33
006 #define MOUSE_LEFT_CLICK   0x01
007
008 #define USHORT unsigned short int
009
010 UCHAR detectMouse(UCHAR *btn);
011 void getMousePos(USHORT *row, USHORT *col, UCHAR *btn);
012 void setMouseRegion(USHORT y1, USHORT x1,
013                      USHORT y2, USHORT x2);
014 void showMouse(void);

```

```

015
016 int main(void)
017 {
018     Screen *layar = new Screen();
019     Keyboard *tombol = new Keyboard(layar);
020
021     USHORT x1, x2, y1, y2;
022     UCHAR status, button, str[5];
023
024     status = detectMouse(&button);
025
026     layar->setMode(0x03);
027     layar->setCursorPos(4, 14);
028     tombol->hideCursor();
029
030     if (!status)
031     {
032         layar->writeString("Mouse tidak siap! Tekan ENTER ...");
033         tombol->getString(str, 0);
034
035         delete tombol; delete layar;
036         exit(EXIT_FAILURE);
037     }
038
039     layar->writeString("Gerakan mouse Anda!");
040     layar->setCursorPos(5, 14);
041     layar->writeString("Klik kiri untuk berhenti.");
042
043     y1 = 9 * 8;      /* Baris ke-10 */
044     y2 = 14 * 8;      /* Baris ke-15 */
045     x1 = 29 * 8;      /* Kolom ke-30 */
046     x2 = 59 * 8;      /* Kolom ke-60 */
047
048     showMouse();          /* Tampilkan pointer mouse */
049     setMouseRegion(y1, x1, y2, x2); /* Batas koordinat mouse */
050
051     while (TRUE)
052     {
053         getMousePos(&y1, &x1, &button);
054
055         if (button == MOUSE_LEFT_CLICK) break;
056     }
057
058     delete tombol; delete layar;
059     return EXIT_SUCCESS;
060 }
061
062 UCHAR detectMouse(UCHAR *btn)
063 {
064     UCHAR state, button;
065
066     asm mov ah, 0x00;      /* Register AH = 0 */
067     asm mov al, 0x00;      /* Register AL = 0, AH:AL = 0:0 */
068     asm int MOUSE_INT;    /* Laksanakan interupsi 0x33 */
069     asm mov state, al;    /* Salin nilai AL ke state */
070     asm mov button, bl;   /* Salin nilai BL ke button */
071

```

```

072     *btn = button;
073     return state;
074 }
075
076 void setMouseRegion(USHORT y1, USHORT x1,
077                      USHORT y2, USHORT x2)
078 {
079     asm mov ax, 0x0007; /* Register AX = 7 */
080     asm mov cx, x1; /* CX = batas horizontal minimal */
081     asm mov dx, x2; /* DX = batas horizontal maksimal */
082     asm int MOUSE_INT; /* Lakukan interupsi 0x33 */
083
084     asm mov ax, 0x0008; /* Register AX = 8 */
085     asm mov cx, y1; /* CX = batas vertikal minimal */
086     asm mov dx, y2; /* DX = batas vertikal maksimal */
087     asm int MOUSE_INT; /* Lakukan interupsi 0x33 */
088
089     return;
090 }
091
092 void getMousePos(USHORT *row, USHORT *col, UCHAR *btn)
093 {
094     USHORT x, y;
095     UCHAR button;
096
097     asm mov ah, 0x00;
098     asm mov al, 0x03;
099     asm int MOUSE_INT;
100    asm mov x, cx;
101    asm mov y, dx;
102    asm mov button, bl;
103
104    *row = y; *col = x;
105    *btn = button;
106
107    return;
108 }
109
110 void showMouse(void)
111 {
112     asm mov ax, 0x000001;
113     asm int MOUSE_INT;
114
115     return;
116 }

```

Pada program contoh24.cpp fungsi yang digunakan untuk membatasi koodinat posisi mouse adalah fungsi setMouseRegion. Fungsi setMouseRegion menerima empat parameter bertipe unsigned integer yang dikirimkan secara nilai. Parameter y1 dan y2 digunakan untuk menentukan batas vertikal, sedangkan parameter x1 dan x2 digunakan untuk menentukan batas horizontal.

4.8. Mengetahui Status Penekanan dan Pelepasan Tombol Mouse

Interupsi 33 heksadesimal juga bisa digunakan untuk mengetahui status penekanan tombol dan pelepasan tombol mouse. Interupsi 33 heksadesimal servis 5

digunakan untuk mengetahui status penekanan tombol mouse, berapa kali tombol tersebut diklik, dan posisi koordinat pointer mouse ketika terakhir kali tombol mouse ditekan serta tombol apa yang diklik. Sedangkan interupsi 33 heksadesimal servis 6 digunakan untuk mengetahui status pelepasan tombol mouse, berapa kali tombol mouse dilepas, posisi koordinat pointer mouse ketika terakhir kali dilepas dan tombol apa yang dilepas. Berikut ini adalah prosedur untuk mengetahui status penekanan dan pelepasan tombol mouse.

Mengetahui status penekanan tombol mouse:

- ☒ Register AH harus bernilai 0.
- ☒ Register AL harus bernilai 5 heksadesimal.
- ☒ Register BX menentukan tombol yang ingin diketahui, jika ingin mengetahui status pelepasan tombol kiri maka nilainya harus 0, tetapi jika yang ingin diketahui adalah tombol kanan maka nilainya harus 1.
- ☒ Lakukan interupsi 33 heksadesimal.

Setelah interupsi 33 heksadesimal servis 5 dilakukan:

- ☒ Register BX menunjukkan banyaknya tombol diklik.
- ☒ Register CX menunjukkan posisi horizontal ketika terakhir kali tombol mouse diklik.
- ☒ Register DX menunjukkan posisi vertikal ketika terakhir kali tombol mouse diklik.

Mengetahui status pelepasan tombol mouse:

- ☒ Register AH harus bernilai 0.
- ☒ Register AL harus bernilai 6 heksadesimal.
- ☒ Register BX menentukan tombol yang ingin diketahui, jika ingin mengetahui status pelepasan tombol kiri maka nilainya harus 0, tetapi jika yang ingin diketahui adalah tombol kanan maka nilainya harus 1.
- ☒ Lakukan interupsi 33 heksadesimal.

Setelah interupsi 33 heksadesimal servis 6 dilakukan:

- ☒ Register BX menunjukkan banyaknya tombol dilepas.
- ☒ Register CX menunjukkan posisi horizontal terakhir kali tombol dilepas.
- ☒ Register DX menunjukkan posisi vertikal terakhir kali tombol dilepas.

Berikut ini adalah contoh program untuk memberikan contoh cara menggunakan interupsi 33 heksadesimal servis 5. Simpan project berikut ini dengan nama contoh25.ide dalam direktori yang sama dengan file screen.cpp dan keyboard.cpp.

contoh25.cpp:

```
01 #include <stdlib.h>
02 #include "screen.cpp"
03 #include "keyboard.cpp"
04
05 #define MOUSE_INT          0x33    /* Nomor interupsi mouse */
06 #define MOUSE_LEFT_BUTTON  0x00    /* Pilihan tombol kiri */
07 #define MOUSE_RIGHT_BUTTON 0x01    /* Pilihan tombol kanan */
08
09 #define USHORT unsigned short int
10
11 UCHAR detectMouse(UCHAR *btn);
12 USHORT getButtonClick(UCHAR btn, USHORT *row,
13                         USHORT *col, USHORT maxclick);
14 void showMouse(void);
15
16 int main(void)
17 {
18     Screen *layar = new Screen();
```

```

19  Keyboard *tombol = new Keyboard(layar);
20
21  USHORT jumlah, baris, kolom;
22  UCHAR status, button, str[5];
23
24  status = detectMouse(&button);
25
26  layar->setMode(0x03); layar->setCursorPos(4, 14);
27  showMouse();
28
29  if (!status)
30  {
31      layar->writeString("Mouse tidak siap! Tekan ENTER ...");
32      tombol->getString(str, 0);
33
34      delete layar; delete tombol;
35      exit(EXIT_FAILURE);
36  }
37
38
39  layar->writeString("Klik kiri 5 kali untuk berhenti");
40  tombol->hideCursor();
41
42  jumlah = getButtonClick(MOUSE_LEFT_BUTTON,
43                         &baris, &kolom, 5);
44
45  delete tombol; delete layar;
46  return EXIT_SUCCESS;
47 }
48
49 UCHAR detectMouse(UCHAR *btn)
50 {
51  UCHAR state, button;
52
53  asm mov ah, 0x00; /* Register AH = 0 */
54  asm mov al, 0x00; /* Register AL = 0, AH:AL = 0:0 */
55  asm int MOUSE_INT; /* Laksanakan interupsi 0x33 */
56  asm mov state, al; /* Salin nilai AL ke state */
57  asm mov button, bl; /* Salin nilai BL ke button */
58
59  *btn = button;
60  return state;
61 }
62
63 USHORT getButtonClick(UCHAR btn, USHORT *row,
64                       USHORT *col, USHORT maxclick)
65 {
66  USHORT click, nclick, x, y;
67
68  click = nclick = x = y = 0;
69
70  while (nclick < maxclick)
71  {
72      asm mov ah, 0x00; /* Register AH = 0 */
73      asm mov al, 0x05; /* Register AL = 0, AH:AL = AX = 0 */
74      asm mov bh, 0x00; /* Register BH = 0 */
75      asm mov bl, btn; /* Register BL = tombol yang dipilih */
76      asm int MOUSE_INT; /* Lakukan interupsi 0x33 */

```

```

77
78     asm mov click, bx;      /* Salin nilai di BX ke click */ 
79     asm mov x, cx;          /* Posisi kolom terakhir */ 
80     asm mov y, dx;          /* Posisi baris terakhir */ 
81
82     nclick += click;
83 }
84
85 *row = y; *col = x;
86
87 return nclick;
88 }
89
90 void showMouse(void)
91 {
92     asm mov ax, 0x00001; /* AH = 0, AL = 1, AX = 1 */
93     asm int MOUSE_INT;   /* Tampilkan pointer mouse */
94
95     return;
96 }
```

Fungsi getButtonClick pada program contoh25.cpp digunakan untuk mendeteksi penekanan tombol sebanyak beberapa kali. Parameter btn yang bertipe unsigned character digunakan untuk menentukan tombol mouse (tombol kiri atau kanan) yang akan dibatasi penekanannya. Parameter col dan row bertipe unsigned integer yang dikirimkan secara referensi digunakan untuk mengetahui posisi baris dan kolom pointer mouse ketika tombol mouse terakhir kali ditekan. Parameter maxclick bertipe unsigned integer digunakan untuk menentukan banyak penekanan tombol mouse.

Setelah mempraktekkan cara mengetahui status penekanan tombol mouse, maka project contoh26.ide berikut ini akan mempraktekan cara mengetahui pelepasan tombol mouse menggunakan interupsi 33 heksadesimal servis 6. Simpan kode program contoh26.cpp dalam direktori yang sama dengan screen.cpp dan keyboard.cpp.

contoh26.cpp:

```

01 #include <stdlib.h>
02 #include "screen.cpp"
03 #include "keyboard.cpp"
04
05 #define MOUSE_INT          0x33    /* Interupsi mouse */
06 #define MOUSE_LEFT_BUTTON  0x00    /* Pilihan tombol kiri */
07 #define MOUSE_RIGHT_BUTTON 0x01    /* Pilihan tombol kanan */
08
09 #define USHORT unsigned short int
10
11 UCHAR detectMouse(UCHAR *btn);
12 USHORT getButtonRelease(UCHAR btn, USHORT *row,
13                         USHORT *col, USHORT maxrel);
14 void showMouse(void);
15
16 int main(void)
17 {
18     Screen *layar = new Screen();
19     Keyboard *tombol = new Keyboard(layar);
20
21     USHORT jumlah, baris, kolom;
```

```

22  UCHAR status, button, str[5];
23
24  status = detectMouse(&button);
25
26  layar->setMode(0x03); layar->setCursorPos(4, 14);
27  showMouse();
28
29  if (!status)
30  {
31      layar->writeString("Mouse tidak siap! Tekan ENTER ...");
32      tombol->getString(str, 0);
33
34      delete layar; delete tombol;
35      exit(EXIT_FAILURE);
36  }
37
38  layar->writeString("Lepas tombol kanan 5x untuk berhenti");
39  tombol->hideCursor();
40
41  jumlah = getButtonRelease(MOUSE_RIGHT_BUTTON,
42                             &baris, &kolom, 5);
43
44  delete tombol; delete layar;
45  return EXIT_SUCCESS;
46 }
47
48 UCHAR detectMouse(UCHAR *btn)
49 {
50     UCHAR state, button;
51
52     asm mov ah, 0x00; /* Register AH = 0 */
53     asm mov al, 0x00; /* Register AL = 0, AH:AL = 0:0 */
54     asm int MOUSE_INT; /* Lakukan interupsi 0x33 */
55     asm mov state, al; /* Salin nilai AL ke state */
56     asm mov button, bl; /* Salin nilai BL ke button */
57
58     *btn = button;
59     return state;
60 }
61
62 USHORT getButtonRelease(UCHAR btn, USHORT *row,
63                         USHORT *col, USHORT maxrel)
64 {
65     USHORT rel, nrel, x, y;
66
67     rel = nrel = x = y = 0;
68
69     while (nrel < maxrel)
70     {
71         asm mov ah, 0x00;
72         asm mov al, 0x06;
73         asm mov bh, 0x00;
74         asm mov bl, btn;
75         asm int MOUSE_INT;
76
77         asm mov rel, bx;
78         asm mov x, cx;

```

```

80     asm mov y, dx;
81
82     nrel += rel;
83 }
84
85 *row = y; *col = x;
86
87 return rel;
88 }
89
90 void showMouse(void)
91 {
92     asm mov ax, 0x00001;
93     asm int MOUSE_INT;
94
95     return;
96 }
```

Fungsi getButtonRelease pada program contoh26.cpp digunakan untuk mendeteksi pelepasan tombol sebanyak beberapa kali. Parameter btr yang bertipe unsigned character digunakan untuk menentukan tombol mouse (tombol kiri atau kanan) yang akan dibatasi pelepasannya. Parameter col dan row bertipe unsigned integer yang dikirimkan secara referensi digunakan untuk mengetahui posisi baris dan kolom pointer mouse ketika tombol mouse terakhir kali dilepas. Parameter maxclick bertipe unsigned integer digunakan untuk menentukan banyak pelepasan tombol mouse.

4.9. Membuat Class untuk Menggunakan Mouse

Setelah memahami dan mempraktekkan teknik-teknik mengoperasikan mouse menggunakan teknik *inline assembly*, maka pada sub bab ini akan dibuat pustaka class (*class library*) untuk mengenkapsulasi fungsi-fungsi operasi mouse yang telah dipraktekkan pada contoh-contoh program sebelumnya. Pustaka class ini akan disimpan dalam file kode program mouse.cpp. Pustaka class.cpp akan bertindak seperti file screen.cpp dan keyboard.cpp yang menyediakan objek untuk operasi mouse. Perhatikan dan pelajari pustaka class mouse.cpp berikut ini, kemudian jelaskanlah letak perbedaan fungsi-fungsi anggota pada class Mouse dengan fungsi-fungsi pada contoh program sebelumnya.

mouse.cpp:

```

001 /*
002  mouse.cpp
003  Class library untuk menggunakan mouse.
004  Hak Cipta Pebi Yudha K.
005  Juni 2009
006
007  Disusun sebagai contoh program
008  Modul Praktikum Pemrograman C++ Lanjutan
009  AMIK BSI
010 */
011
012 #define MOUSE_INT      0x33 /* Interupsi mouse      */
013 #define MOUSE_READY    0xff /* Mouse siap digunakan */
014 #define MOUSE_NOT_READY 0x00 /* Mouse tidak siap      */
015 #define MOUSE_SUCCESS   0x1f /* Bisa dinonaktifkan   */
016 #define MOUSE_FAILURE    0xff /* Tak bisa dinonaktifkan */
017 #define MOUSE_LEFT_CLICK 0x01 /* Klik tombol kiri      */
018 #define MOUSE_RIGHT_CLICK 0x02 /* Klik tombol kanan     */
```

```

019 #define MOUSE_LEFT_BUTTON 0x00 /* Pilih tombol kiri */
020 #define MOUSE_RIGHT_BUTTON 0x01 /* Pilih tombol kanan */
021
022 #define UCHAR unsigned char
023 #define USHORT unsigned short int
024
025 class Mouse
026 {
027     private:
028         UCHAR button; /* Banyaknya tombol mouse */
029         UCHAR isReady; /* Status mouse siap/tidak */
030         UCHAR clickMode; /* Klik kiri atau klik kanan */
031         USHORT x, y; /* Posisi koordinat mouse */
032
033     /* Mengetahui posisi koordinat pointer mouse */
034     void getMousePos(void);
035
036     public:
037     Mouse(void); /* Konstruktor default */
038     ~Mouse(void); /* Destruktor default */
039
040     UCHAR getState(void); /* Mouse siap/tidak siap */
041     UCHAR getNumButton(void); /* Cek jumlah tombol mouse */
042
043     UCHAR disableMouse(void); /* Nonaktifkan driver mouse */
044     void enableMouse(void); /* Aktifkan driver mouse */
045
046     /* Mendeteksi penekanan tombol mouse */
047     USHORT getButtonClick(UCHAR btn, USHORT maxclick);
048
049     /* Mendeteksi pelepasan tombol mouse */
050     USHORT getButtonRelease(UCHAR btn, USHORT maxrel);
051
052     /* Mengetahui klik kiri atau klik kanan setelah */
053     /* fungsi getMousePos dijalankan */
054     UCHAR getClickMode(void);
055
056     USHORT getX(void); /* Posisi horizontal mouse */
057     USHORT getY(void); /* Posisi vertikal mouse */
058
059     void hideMouse(void); /* Menyembunyikan pointer */
060     void showMouse(void); /* Menampilkan pointer */
061
062     /* Memindahkan pointer mouse di posisi tertentu */
063     void setMousePos(USHORT row, USHORT col);
064
065     /* Membatasi posisi koordinat mouse di layar */
066     void setMouseRegion(USHORT y1, USHORT x1,
067                         USHORT y2, USHORT x2);
068 };
069
070 Mouse::Mouse(void)
071 {
072     UCHAR state, button;
073
074     asm mov ah, 0x00; /* Register AH = 0 */
075     asm mov al, 0x00; /* Register AL = 0, AH:AL = 0:0 */

```

```

076     asm int MOUSE_INT;      /* Deteksi mouse dan drivernya */
077     asm mov state, al;      /* Salin nilai AL ke state */
078     asm mov button, bl;      /* Salin nilai BL ke button */
079
080     this->isReady = state;
081     this->button = button;
082
083     return;
084 }
085
086 Mouse::~Mouse(void)
087 {
088     return;
089 }
090
091 UCHAR Mouse::getState(void)
092 {
093     return this->isReady;
094 }
095
096 UCHAR Mouse::getNumButton(void)
097 {
098     return this->button;
099 }
100
101 UCHAR Mouse::disableMouse(void)
102 {
103     UCHAR state;
104
105     asm mov ah, 0x00;      /* Register AH = 0 */
106     asm mov al, 0x1f;      /* Register AL = 0x1f */
107     asm int MOUSE_INT;    /* Nonaktifkan driver mouse */
108     asm mov state, al;      /* Salin register AL ke state */
109
110     return state;
111 }
112
113 void Mouse::enableMouse(void)
114 {
115     asm mov ax, 0x0020;    /* AH = 0, AL = 0x20 */
116     asm int MOUSE_INT;    /* Aktifkan driver mouse */
117
118     return;
119 }
120
121 USHORT Mouse::getButtonClick(UCHAR btn, USHORT maxclick)
122 {
123     USHORT click, nclick, x, y;
124
125     click = nclick = x = y = 0;
126
127     while (nclick < maxclick)
128     {
129         asm mov ah, 0x00;      /* Register AH = 0 */
130         asm mov al, 0x05;      /* Register AL = 5, AX = 5 */
131         asm mov bh, 0x00;      /* Register BH = 0 */
132         asm mov bl, btn;      /* BL = tombol yang dipilih */

```

```

136     asm int MOUSE_INT; /* Deteksi status klik tombol */
137
138     asm mov click, bx; /* Banyaknya penekanan tombol */
139     asm mov x, cx; /* Posisi horizontal mouse */
140     asm mov y, dx; /* Posisi vertikal mouse */
141
142     nclick += click;
143 }
144
145 this->y = y; this->x = x;
146
147 return click;
148 }
149
150 USHORT Mouse::getButtonRelease(UCHAR btn, USHORT maxrel)
151 {
152     USHORT rel, nrel, x, y;
153
154     rel = nrel = x = y = 0;
155
156     while (nrel < maxrel)
157     {
158         asm mov ah, 0x00; /* Register AH = 0 */
159         asm mov al, 0x06; /* Register AL = 6, AX = 6 */
160         asm mov bh, 0x00; /* Register BH = 0 */
161         asm mov bl, btn; /* BL = tombol yang dipilih */
162         asm int MOUSE_INT; /* Deteksi pelepasan tombol */
163
164         asm mov rel, bx; /* Banyaknya pelepasan tombol */
165         asm mov x, cx; /* Posisi horizontal mouse */
166         asm mov y, dx; /* Posisi vertikal mouse */
167
168         nrel += rel;
169     }
170
171     this->y = y; this->x = x;
172
173     return rel;
174 }
175
176 void Mouse::getMousePos(void)
177 {
178     USHORT x, y;
179     UCHAR cmode;
180
181     asm mov ax, 0x0003; /* AH = 0, AL = 3, AX = 3 */
182     asm int MOUSE_INT; /* Deteksi posisi mouse */
183     asm mov x, cx; /* Posisi horizontal mouse */
184     asm mov y, dx; /* Posisi vertikal mouse */
185     asm mov cmode, bl; /* Tombol yang diklik */
186
187     this->y = y; this->x = x;
188     this->clickMode = cmode;
189
190     return;
191 }
192

```

```

193 UCHAR Mouse::getClickMode(void)
194 {
195     this->getMousePos();
196     return this->clickMode;
197 }
198
199 USHORT Mouse::getX(void)
200 {
201     return this->x;
202 }
203
204 USHORT Mouse::getY(void)
205 {
206     return this->y;
207 }
208
209 void Mouse::hideMouse(void)
210 {
211     asm mov ax, 0x0002;      /* AH = 0, AL = 2, AX = 2      */
212     asm int MOUSE_INT;      /* Sembunyikan pointer mouse */
213
214     return;
215 }
216
217
218 void Mouse::setMousePos(USHORT row, USHORT col)
219 {
220     asm mov ax, 0x0004;      /* AH = 0, AL = 4, AX = 4      */
221     asm mov cx, col;         /* Posisi horizontal mouse    */
222     asm mov dx, row;         /* Posisi vertikal mouse      */
223     asm int MOUSE_INT;      /* Pindahkan pointer mouse    */
224
225     return;
226 }
227
228
229 void Mouse::setMouseRegion(USHORT y1, USHORT x1,
230                             USHORT y2, USHORT x2)
231 {
232     asm mov ax, 0x0007;      /* Register AX = 7            */
233     asm mov cx, x1;          /* CX = batas horizontal minimal */
234     asm mov dx, x2;          /* DX = batas horizontal maksimal */
235     asm int MOUSE_INT;      /* Batasi posisi horizontal mouse */
236
237     asm mov ax, 0x0008;      /* Register AX = 8            */
238     asm mov cx, y1;          /* CX = batas vertikal minimal */
239     asm mov dx, y2;          /* DX = batas vertikal maksimal */
240     asm int MOUSE_INT;      /* Batasi posisi vertikal mouse */
241
242     return;
243 }
244
245
246 void Mouse::showMouse(void)
247 {
248     asm mov ax, 0x0001;      /* AH = 0, AL = 1, AX = 1      */
249     asm int MOUSE_INT;      /* Tampilkan pointer mouse    */
250
251     return;
252 }
253
254
255 
```

Setelah membuat pustaka class untuk operasi mouse, berikut ini akan diberikan contoh cara menggunakan pustaka class mouse.cpp. Objek yang diharus diinstansiasikan dari pustaka class mouse.cpp adalah objek Mouse. Program berikut ini adalah program untuk mendeteksi pilihan yang ditampilkan dilayar. Pada layar monitor akan ditampilkan pilihan A, B, dan C. Jika pengguna mengklik A atau B maka muncul pesan bahwa pengguna telah memilih A atau B. Tetapi jika pengguna mengklik C maka program akan dihentikan. Simpan project contoh27.ide berikut ini dalam direktori yang sama dengan file screen.cpp, keyboard.cpp dan mouse.cpp.

contoh27.cpp:

```
01 #include <stdlib.h>
02 #include "screen.cpp"
03 #include "keyboard.cpp"
04 #include "mouse.cpp"
05
06 int main(void)
07 {
08     Screen *layar = new Screen();
09     Keyboard *tombol = new Keyboard(layar);
10     Mouse *tikus = new Mouse();
11
12     UCHAR str[5];
13     USHORT x, y;
14
15     layar->setMode(0x03); layar->setCursorPos(4, 14);
16     tombol->hideCursor();
17
18     if (tikus->getState() == MOUSE_NOT_READY)
19     {
20         layar->writeString("Mouse tidak siap! Tekan ENTER ...");
21         tombol->getString(str, 0);
22     }
23
24     tikus->showMouse();
25     layar->writeString("Klik A, B atau C");
26     layar->setCursorPos(5, 14);
27     layar->writeString("[A] Huruf A");
28     layar->setCursorPos(6, 14);
29     layar->writeString("[B] Huruf B");
30     layar->setCursorPos(7, 14);
31     layar->writeString("[C] Selesasi");
32
33     while (TRUE)
34     {
35         layar->setCursorPosition(9, 14);
36
37         if (tikus->getClickMode() == MOUSE_LEFT_CLICK)
38         {
39             x = tikus->getX() / 8; /* Periksa posisi horizontal */
40             y = tikus->getY() / 8; /* Periksa posisi vertikal */
41
42             if ((x == 15) && (y == 5))
43                 layar->writeString("Anda memilih A!");
44
45             if ((x == 15) && (y == 6))
46                 layar->writeString("Anda memilih B!");
```

```

47         if ((x == 15) && (y == 7)) break;
48     }
49 }
50 }
51 delete layar; delete tombol; delete tikus;
52 return EXIT_SUCCESS;
53 }
54 }
```

4.10. Latihan-latihan Bab IV

1. Jelaskan cara atau prosedur yang harus dilakukan untuk mendeteksi mouse apakah siap atau tidak siap digunakan!
2. Ubah program contoh18.cpp yang dibuat dengan teknik inline assembly menjadi menggunakan fungsi standar int16 dan jalankan!
3. Jelaskan prosedur untuk menampilkan dan menyembunyikan pointer mouse menggunakan diagram alur program (*flow chart*)!
4. Ubah program contoh19.cpp yang dibuat dengan teknik inline assembly menjadi menggunakan fungsi standar int16 dan jalankan!
5. Buat sebuah program yang dapat menampilkan posisi koordinat mouse, status tombol yang ditekan serta bisa mengaktifkan atau menonaktifkan driver mouse menggunakan int86!
6. Dengan memanfaatkan pustaka class mouse.cpp, screen.cpp dan keyboard.cpp buatlah sebuah program dengan tampilan seperti dibawah ini!

Silahkan masukan username dan password Anda

Username : []
 Password : []

[LOGIN] [BATAL] [EXIT]

Jika [LOGIN] diklik muncul pesan:

Selamat datang <username>, password Anda adalah <password>.

Jika [BATAL] diklik maka username dan password akan dikosongkan dan pointer mouse fokus di username.

Jika [EXIT] diklik maka program berhenti.

7. Ubah program contoh23.cpp yang dibuat dengan teknik inline assembly menjadi menggunakan fungsi int86 dan jalankan!
8. Jelaskan algoritma program contoh23.cpp menggunakan diagram alur program!
9. Buat sebuah program sederhana untuk menjalankan interupsi 33 heksadesimal servis 5!
10. Buat sebuah program dengan rancangan input dan output seperti dibawah ini dengan memanfaatkan pustaka class screen.cpp, keyboard.cpp dan mouse.cpp!

Kalkulator Sederhana Menggunakan C++

```
Nilai 1: [ ] Operasi Matematika:  
Nilai 2: [ ] [ + ] [ X ] [ % ] [ ! ]  
Hasil : [ ] [ - ] [ / ] [ ^ ] [ C ]  
[ Q ]
```

Keteterangan tombol:

- [+] Menghitung penjumlahan
- [-] Menghitung pengurangan
- [X] Menghitung perkalian
- [/] Menghitung pembagian
- [%] Menghitung sisa pembagian (modulus)
- [^] Menghitung pangkat (power)
- [!] Menghitung faktorial
- [C] Membersihkan nilai 1, nilai 2, dan hasil
- [Q] Menghentikan program