Computer Science | Udayana University

PEMROGRAMAN KOMPUTER

C Language Series

MODUL 4

(Pointer dan Fungsi)

Tim Pengajar Algoritma dan Pemrograman

Nama Mata Kuliah	: Praktikum Pemrograman Komputer
Bobot sks	: 1 SKS
Semester	: II (Dua)
Koordinator MK	: I Made Widiartha, S.Si.,M.Kom
Modul Praktikum	: Modul 4 – Pointer dan Fungsi

1. Deskripsi

Pada praktikum ini akan dipraktekan tentang konsep pointer dan fungsi pada bahasa pemrograman C yang meliputi: Fungsi-fungsi dasar pada bahasa pemrograman C, Fungsi Rekursif, dan Pointer.

2. Tujuan Umum

Setelah mengikuti praktikum ini, mahasiswa mampu menerapkan konsep fungsi dan menggunakan variabel dalam bentuk pointer pada sebuah program.

3. Indikator Pencapaian

Adapun indikator pencapaian dari modul praktikum ini adalah maha siswa dapat mengimplementasikan konsep pointer dan fungsi dalam penyelesaian permasalahan - permasalahan yang ada ke dalam sebuah program sederhana.

4. Teori

4.1. Pointer

4.1.1. Pengertian Pointer

Pointer (variabel penunjuk) adalah suatu variabel yang berisi alamat memori dari suatu variabel lain. Alamat ini merupakan lokasi dari obye k lain (biasanya variabel lain)

di dalam memori. Contoh, jika sebuah variabel berisi alam at dari variabel lain, variable pertama dikatakan menunjuk ke variabel kedua. Operator Pointer ada dua, yaitu :

1. Operator &
    - Operator & bersifat unary (hanya memerlukan satu operand saja).
    - Operator & menghasilkan alamat dari operandnya.
2. Operator *
    - Operator * bersifat unary (hanya memerlukan satu operand saja).
    - Operator * menghasilkan nilai yang berada pada sebuah alamat.

4.1.2. Deklarasi Pointer

Seperti halnya variabel yang lain, variabel pointer juga haru s dideklarasikan terlebih dahulu sebelum digunakan. Bentuk Umum :

```
Tipe_data *nama_pointer;
```

Tipe data pointer mendefinisikan tipe dari ob yek yang ditunjuk oleh pointer. Secara teknis, tipe apapun dari pointer dapat menunjukkan lokasi (dimanapun) dalam memori. Bahkan operasi pointer dapat dilaksanakan relatif terhadap tipe dasar apapun yang ditunjuk. Contoh, ketika kita mendeklarasikan pointer dengan tipe int*, compiler akan menganggap alamat yang ditunjuk menyimpan nilai integer - walaupun sebenarnya bukan (sebuah pointer int* selalu menganggap bahwa ia menunjuk ke sebuah obyek bertipe integer, tidak peduli isi sebenarnya). Karenanya, sebelum mendeklarasikan sebuah pointer, pastikan tipenya sesuai dengan tipe obyek yang akan ditunjuk.

Contoh :

```
int *px;
char *sh;
```

```
#include "stdio.h"
#include "conio.h"

void main(){

      int x, y; // x dan y bertipe int
      int *px; // px pointer yang menunjuk objek

      x = 87;
      px = &x; /* px berisi alamat dari x */
      y = *px; /* y berisi nilai yang ditunjuk px */

      printf("Alamat x = %p\n", &x);
      printf("Isi px = %p\n", px);
      printf("Isi x = %i\n", x);
      printf("Nilai yang ditunjuk oleh px = %i\n", *px);
      printf("Nilai y = %i\n", y);

      getch();
}
```

4.1.3. Operasi Pointer

- Operasi Penugasan

  Suatu variable pointer seperti halnya variable yang lain, juga bias mengalami operasi penugasan. Nilai dari suatu variable pointer dapat disalin ke variable pointer yang lain.

- Operasi Aritmatika

  Suatu variabel pointer hanya dapat dila kukan operasi aritmatika dengan nilai integer saja. Operasi yang biasa dilakukan adalah operasi penambahan dan pengurangan. Operasi penambahan dengan suatu nilai menunju kkan lokasi data berikutnya (index selanjutnya) dalam memori. Begitu juga operasi pengurangan.

- Operasi Logika

  Suatu pointer juga dapat dikenai operasi logika.

```
#include "stdio.h"
#include "conio.h"

void main(){

	//================ Operasi Penugasan ================================
	float *x1, *x2, y;

	y = 13.45;

	x1 = &y; // Alamat dari y disalin ke variabel x1
	x2 = x1; // Isi variabel x1 disalin ke variabel x2

	printf("Nilai variabel y = %.2f ada di alamat %p\n", y, x1);
	printf("Nilai variabel y = %.2f ada di alamat %p\n", y, x2);


	//===================================================================



	//================ Operasi Aritmatika ================================

	int nilai[3], *penunjuk;

	nilai[0] = 125;
	nilai[1] = 345;
	nilai[2] = 750;

	penunjuk = &nilai[0];

	printf("Nilai %i ada di alamat memori %p\n", *penunjuk, penunjuk);
	printf("Nilai %i ada di alamat memori %p\n", *(penunjuk+1), penunjuk+1);
	printf("Nilai %i ada di alamat memori %p\n", *(penunjuk+2), penunjuk+2);

	//===================================================================



	//==================== Operasi Logika ================================

	int a = 100, b = 200, *pa, *pb;

	pa = &a;
	pb = &b;

	if(pa < pb) printf("pa menunjuk ke memori lebih rendah dari pb\n");
	if(pa == pb) printf("pa menunjuk ke memori yang sama dengan pb\n");
	if(pa > pb) printf("pa menunjuk ke memori lebih tinggi dari pb\n");

	//===================================================================


	getch();

}
```

4.1.4. Pointer dan String

```
#include "stdio.h"
#include "conio.h"

char *nama1 = "SPIDERMAN";
char *nama2 = "GATOTKACA";

void main(){

      char namax;

      printf("SEMULA :");
      printf("Saya suka >> %s\n", nama1);
      printf("Tapi saya juga suka >> %s\n", nama2);

      // Penukaran string yang ditunjuk oleh pointer nama1 dan nama2
      printf("SEKARANG :");
      printf("Saya suka >> %s\n", nama1);
      printf("Dan saya juga masih suka >> %s\n", nama2);

      getch();
}
```

4.1.5. Pointer Menunjuk Suatu Array

```
#include "stdio.h"
#include "conio.h"

void main(){

      static int tgl_lahir[] = { 13,9,1982 };

      int *ptgl;

      ptgl = tgl_lahir; // ptgl berisi alamat array

      printf("Diakses dengan pointer");
      printf("Tanggal = %i\n", *ptgl);
      printf("Bulan = %i\n", *(ptgl + 1));
      printf("Tahun = %i\n", *(ptgl + 2));
      printf("\nDiakses dengan array biasa\n");
      printf("Tanggal = %i\n", tgl_lahir[0]);
      printf("Bulan = %i\n", tgl_lahir[1]);
      printf("Tahun = %i\n", tgl_lahir[2]);

      getch();
}
```

4.1.6. Memberi Nilai Array dengan Pointer

```
#include "stdio.h"
#include "conio.h"

void main(){

      int x[5], *p, k;

      p = x;

      x[0] = 5; // x[0] diisi dengan 5 sehingga x[0] = 5

      x[1] = x[0]; // x[1] diisi dengan x[0] sehingga x[1] = 5

      x[2] = *p + 2; // x[2] diisi dengan x[0] + 2 sehingga x[2] = 7

      x[3] = *(p+1) - 3; // x[3] diisi dengan x[1] - 3 sehingga x[3] = 2

      x[4] = *(x + 2); // x[4] diisi dengan x[2] sehingga x[4] = 7

      for(k=0; k<5; k++){
            printf("x[%i] = %i\n", k, x[k]);
      }

      getch();
}
```

4.2. Fungsi

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi `main()`. Fungsi banyak diterapkan dalam program program C yang terstruktur. Keuntungan penggunaan fungsi dalam program yaitu program akan memiliki struktur yang jelas dan juga akan menghindari penulisan bagian program yang sama.

Dalam bahasa C fungsi dapat dibagi menjadi dua, yaitu fungsi pustaka atau fungsi yang telah tersedia dalam Bahasa Pemrograman C dan fungsi yang didefinisikan atau dibuat oleh programmer.

4.2.1. Beberapa Fungsi Pustaka pada Bahasa Pemrograman C

Fungsi Operasi String (tersimpan dalam header file "string.h")

- strcpy()
  - Berfungsi untuk menyalin suatu string asal ke variable string tujuan.
  - Bentuk umum : `strcpy(var_tujuan, string_asal);`
- strlen()
  - Berfungsi untuk memperoleh jumlah karakter dari suatu string.
  - Bentuk umum : `strlen(string);`
- strcat()
  - Digunakan untuk menambahkan string sumber ke bagian akhir dari string tujuan.
  - Bentuk umum : `strcat(tujuan, sumber);`
- strupr()
  - Digunakan untuk mengubah setiap huruf dari suatu string menjadi huruf capital.
  - Bentuk umum : `strupr(string);`
- strlwr()
  - Digunakan untuk mengubah setiap huruf dari suatu string menjadi huruf kecil semua.
  - Bentuk umum : `strlwr(string);`

- strcmp()
  - Digunakan untuk membandingkan dua buah string.
  - Hasil dari fungsi ini bertipe integer dengan nilai :
    (a) Negative, jika string pertama kurang dari string k edua.
    (b) Nol, jika string pertama sama dengan string kedua
    (c) Positif, jika string pertama lebih besar dari string kedua.
  - Bentuk umum : `strcmp(string1, string2);`

Fungsi Operasi Karakter (tersimpan dalam header "ctype.h")

♦ islower()

- Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kecil.
- Bentuk umum : `islower(char);`

♦ isupper()

- Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kapital.
- Bentuk umum : `isupper(char);`

♦ isdigit()

- Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan sebuah digit.
- Bentuk umum : `isdigit(char);`

♦ tolower()

- Fungsi akan mengubah huruf capital menjadi huruf kecil.
- Bentuk umum : `tolower(char);`

♦ toupper()

- Fungsi akan mengubah huruf kecil menjadi huruf kapital.
- Bentuk umum : `toupper(char);`

Fungsi Operasi Matematik (tersimpan dalam header "math.h" dan "stdlib.h")

♦ sqrt()

- Digunakan untuk menghitung akar dari sebuah bilangan.
- Bentuk umum : `sqrt(bilangan);`

♦ pow()

- Digunakan untuk menghitung pemangkatan suatu bilangan.
- Bentuk umum : `pow(bilangan, pangkat);`

♦ sin(), cos(), tan()

- Masing-masing digunakan untuk menghitung nilai sinus, cosinus dan tangens dari suatu sudut.
- Bentuk umum :

```
sin(sudut);
cos(sudut);
tan(sudut);
```

♦ atof()

- Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe double.
- Bentuk umum : `atof(char x);`

♦ atoi()

- Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe integer.
- Bentuk umum : `atoi(char x);`

♦ div()

- Digunakan untuk menghitung hasil pembagian dan sisa pembagian.
- Bentuk umum : `div_t div(int x, int y)`
- Strukturnya :

```
typedef struct{
     int qout; // hasil pembagian
     int rem // sisa pembagian
} div_t;
```

♦ max()

- Digunakan untuk menentukan nilai maksimal dari dua buah bilangan.
- Bentuk umum : `max(bilangan1, bilangan2);`

♦ min()

- Digunakan untuk menentukan bilangan terkecil dari dua buah bilangan.
- Bentuk umum : `min(bilangan1, bilangan2);`

4.2.2. Membuat Fungsi Sendiri

Deklarasi Fungsi

Sebelum digunakan (dipanggil), suatu fungsi harus dideklarasikan dan didefinisikan terlebih dahulu. Bentuk umum pendeklarasian fungsi adalah :

```
tipe_fungsi nama_fungsi(parameter_fungsi);
```

Sedangkan bentuk umum pendefinisian fungsi adalah :

```
Tipe_fungsi nama_fungsi(parameter_fungsi){
      statement
      statement
      ...
      ...
}
```

Hal-hal yang perlu diperhatikan dalam penggunaan fungsi :

- Kalau tipe fungsi tidak disebutkan, maka akan dianggap sebagai fungis dengan nilai keluaran bertipe integer.
- Untuk fungsi yang memiliki keluaran bertipe bukan integer, maka diperlukan pendefinisian penentu tipe fungsi.
- Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void
- Pernyataan yang diberikan untuk memberikan nilai akhir fun gsi berupa pernyataan return.
- Suatu fungsi dapat menghasilkan nilai balik bagi fungsi pemanggilnya.

```
#include "stdio.h"
#include "conio.h"

// prototype fungsi tambah(), ada titik koma
float tambah(float x, float y);


void main(){
      float a, b, c;
      printf("A = "); scanf("%f", &a);
      printf("B = "); scanf("%f", &b);
      c = tambah(a, b); // pemanggilan fungsi tambah()
      printf("A + B = %.2f", c);
      getch();
}


// Definisi fungsi , tanpa titik koma
float tambah(float x, float y){
      return (a+b); // Nilai balik fungsi
}
```

Parameter Formal dan Parameter Aktual

- Parameter Formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
- Parameter Aktual adalah variabel (parameter) yang dipakai dalam pemanggilan fungsi.

Dalam contoh program pertambahan di atas parameter formal terdapat pada pendefinisisan fungsi :

```
float tambah(float x, float y) //parameter formal
{
   return (a+b);
}
```

Sedangkan parameter aktual terdapat pada pemanggilan fungsi :

```
void main(){
   ...
   c = tambah(a, b); //parameter aktual
   ...
}
```

Cara Melewatkan Parameter

Cara melewatkan suatu parameter dalam Bahasa C ada dua cara yaitu :

1. Pemanggilan Secara Nilai (Call by Value)
   - Call by value akan menyalin nilai dari parameter aktual ke parameter formal.
   - Yang dikirimkan ke fungsi adalah nilai dari datanya, bukan alamat memori letak dari datanya.
   - Fungsi yang menerima kiriman nilai akan menyimpannya di alamat terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi.
   - Perubahan nilai di fungsi (parameter formal) tidak akan merubah nilai asli di bagian program yang memanggilnya.
   - Pengiriman parameter secara nilai adalah pengiriman searah, yaitu dari bagian program yang memanggil fungsi ke fungsi yang dipanggil.
   - Pengiriman suatu nilai dapat dilakukan untuk suatu ungkapan, tidak hanya untuk sebuah variabel, elemen array atau konstanta saja.

```
#include "stdio.h"
#include "conio.h"

// pendeklarasian fungsi
void tukar(int x, int y);

void main(){
      int a,b;
      a = 15;
      b = 10;
      printf("Nilai sebelum pemanggilan fungsi\n");

      // a dan b sebelum pemanggilan fungsi
      printf("a = %i b = %i\n\n", a, b);

      // pemanggilan fungsi tukar()
      tukar(a,b);

      printf("Nilai setelah pemanggilan fungsi\n");

      // a dan b setelah pemanggilan fungsi
      printf("a = %i b = %i\n\n", a, b);

      getch();
}


// Pendefinisian fungsi tukar()
void tukar(int x, int y){

      int z; // variabel sementara

      z = x;
      x = y;
      y = z;

      printf("Nilai di akhir fungsi tukar()\n");
      printf("x = %i y = %i\n\n", x, y);
}
```

2. Pemanggilan Secara Referensi (Call by Reference)
   - Pemanggilan secara Referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.
   - Yang dikirimkan ke fungsi adalah alamat letak dari nilai datanya, bukan nilai datanya.

- Fungsi yang menerima kiriman alamat ini makan menggunakan alamat yang sama untuk mendapatkan nilai datanya.
- Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil fungsi.
- Pengiriman parameter secara referensi adalah pengiriman dua arah, yaitu dari fungsi pemanggil ke fungsi yang dipanggil dan juga sebaliknya.
- Pengiriman secara acuan tidak dapat bdilakukan untuk suatu ungkapan.

```
#include "stdio.h"
#include "conio.h"

void tukar(int *px, int *py);

void main(){
      int a,b;
      a = 15;
      b = 10;

      printf("Nilai sebelum pemanggilan fungsi\n");
      printf("a = %i b = %i\n\n", a, b);

      // parameter alamat a dan alamat b
      tukar(&a,&b);


      printf("Nilai setelah pemanggilan fungsi\n");
      printf("a = %i b = %i\n\n", a, b);

      getch();
}


void tukar(int *px, int *py){

      int z; // variabel sementara

      z = *px;
      *px = *py;
      *py = z;

      printf("Nilai di akhir fungsi tukar()\n");
      printf("*px = %i *py = %i\n\n", *px, *py);
}
```

Penggolongan Variabel berdasarkan Kelas Penyimpanan (Storage Class)

1. Variabel lokal

Variabel lokal adalah variabel yang dideklarasikan di dalam fungsi. Sifat-sifat variabel lokal :

- Secara otomatis akan diciptakan ketika fungsi dipanggil dan akan lenyap ketika proses eksekusi terhadap fungsi berakhir.
- Hanya dikenal oleh fungsi tempat variabel dideklarasikan
- Tidak ada inisialisasi secara otomatis (saat variabel diciptakan nilainya random).
- Dideklarasikan dengan menambahkan kata "auto" (opsional).

2. Variabel global (eksternal)

Variabel global (eksternal) adalah variabel yang dideklarasikan di luar fungsi. Sifat-sifat variabel global :

- Dikenal (dapat diakses) oleh semua fungsi.
- Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
- Dideklarasikan dengan menambahkan kata "extern" (opsional).

```
#include "stdio.h"
#include "conio.h"

void tampil(void);

int i = 25; // variabel global

void main(){

      printf("Nilai variabel i dalam fungsi main() adalah %i\n\n", i);

      i = i * 4; // nilai i yang dikali 4 adalah 25 (global) bukan 10

      printf("\nNilai variabel i dalam fungsi main() sekarang adalah %i\n\n", i);

      getch();
}

void tampil(void){

      int i = 10; // variabel local

      printf("Nilai variabel i dalam fungsi tampil() adalah %i\n\n", i);
}
```

3. Variabel Statis

Variabel statis adalah variabel yang nilainya tetap dan bisa berupa variabel local (internal) dan variabel global (eksternal). Sifat-sifat variabel statis :

- Jika bersifat internal (lokal), maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Jika bersifat eksternal (global), maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada program yang sama.
- Nilai variabel statis tidak akan hilang walau eksekusi terhadap fungsi telah berakhir.
- Inisialisasi hanya perlu dilakukan sekali saja, yaitu pada saat fungsi dipanggil pertama kali.
- Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
- Dideklarasikan dengan menambahkan kata " static".

4. Variabel Register

Variabel Register adalah variabel yang nilainya disimpan dalam register dan bukan dalam memori RAM. Sifat-sifat variabel register :

- Hanya dapat diterapkan pada variabel lokal yang bertipe int dan char.
- Digunakan untuk mengendalikan proses perulangan (looping).
- Proses perulangan akan lebih cepat karena variabel register memiliki kecepatan yang lebih tinggi dibandingkan variabel biasa.
- Dideklarasikan dengan menambahkan kata " register".

```
#include "stdio.h"
#include "conio.h"

void main(){

      register int i; // variabel register

      int jumlah;

      for(i=1; i<=100; i++){
            jumlah = jumlah + i;
      }

      printf("1+2+3+...+100 = %i\n", jumlah);

      getch();
}
```

Fungsi Rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri.

```
#include "stdio.h"
#include "conio.h"

long int faktorial(int N); // prototype fungsi faktorial

void main(){

      int N;

      printf("Berapa factorial ? "); scanf("%i", &N);
      printf("Faktorial dari %i = %ld\n", N, faktorial(N));

      getch();
}


// definisi fungsi factorial
long int faktorial(int N) {

      if(N==0) return(1);

      else{
            // memanggil fungsi faktorial()fungsi faktorial()
            return(N * faktorial(N - 1));
      }

}
```

5. Referensi

1. Munir, R. 1999. Algoritma dan Pemrograman Dalam Bahasa Pascal dan C. Bandung: Informatika.
2. Kadir, A dan Heriyanto. 2005. Algoritma Pemrograman Menggunakan C++. Yogyakarta: Penerbit Andi.
3. Tosin, R. 1997. Flowchart untuk Siswa dan Mahasiswa. Jakarta: DINASTINDO.

6. Pelaksanaan Praktikum

6.1. Tools

Tools yang dapat digunakan pada untuk menunjang praktikum kali ini antara lain:

1. Code :: Block IDE sebagai IDE bahasa pemrograman C

6.3. Langkah-langkah Praktikum

Berikut akan dipaparkan langkah-langkah pembuatan program sederhana dengan mengimplementasikan penggunaan fungsi dan konsep pointer pada bahasa pemrograman C.

Studi kasus :

Mr. X adalah seorang mata-mata di kepolisian. Kali ini dia mendapat tugas untuk memata-matai gembong narkoba kelas kakap. Setiap hari Mr.X harus melaporkan kegiatan gembong yang diintainya kepada atasannya dalam bentuk SMS (Short Message Service) yang sisandikan. Untuk memastikan bahwa pesan yang disampaikan benar-benar dari Mr.X, maka atasan Mr.X membuat aturan penulisan pesan yang harus digunakan oleh Mr.X pada setiap pengiriman pesannya. Mr.X hanya boleh mengirimkan 1 buah pesan setiap harinya, jika lebih dari itu maka operasi ini telah terbongkar dan ada kemungkinan Mr.X telah ditangkap.

Aturan penulisan pesan yang digunakan adalah setiap pesan yang dikirim harus sepanjang 160 karakter. Penyusunan pesan tersebut dilakukan membentuk matriks spiral yang dimulai pusat matriks 1 karakter pertama, lalu 1 karakter berikutnya ke kanan, lalu 1 karakter berikutnya ke bawah, lalu 2 karakter berikutnya ke kiri, lalu 2 karakter berikutnya ke atas, 3 karakter berikutnya ke kanan, 3 karakter berikutnya ke bawah, dan seterusnya hingga semua karakter dalam kalimat termasuk dalam spiral. Khususnya, karakter spasi di ganti dengan "_" (underscore), dan jika ada baris/kolom tersisa setelah karakter terakhir maka elemen-elemen matriks diisi juga dengan "_" (underscore) tersebut. Misalnya kalimat "Hari ini mereka mengirim barang dengan mobil pick-up pukul 12.10 ke jalan nirmala no.22, pukul 15.30 ke jalan asoka, dan pukul 18.02 ke jalan mahendradata" Dikodekan kedalam matriks sebagai berikut:

_	_	_	_									
_	a	s	o	k	a	,	_	d	a	n	_	p
_	_	n	_	n	i	r	m	a	l	a	_	u
a	n	a	i	l	_	p	i	c	k	-	n	k
t	a	l	b	i	r	i	m	_	b	u	o	u
a	l	a	o	g	n	i	_	m	a	p	.	l
d	a	j	m	n	i	H	a	e	r	_	2	_
a	j	_	_	e	_	i	r	r	a	p	2	1
r	_	e	n	m	_	a	k	e	n	u	,	8
d	e	k	a	g	n	e	d	_	g	k	_	.
n	k	_	0	1	.	2	1	_	l	u	p	0
e	_	0	3	.	5	1	_	l	u	k	u	2
h	a	m	_	n	a	l	a	j	_	e	k	_

Sehingga nantinya pesan yang dikirim Mr.X kedapa atasannya adalah sebagai berikut:

```
_ _ _ _ _ a s o k a , _ d a n _ p _ _ n _ n i r
m a l a _ u a n a i l _ p i c k - n k t a l b i
r i m _ b u o u a l a o g n I _ m a p . l d a j
m n i H a e r _ 2 _ a j _ _ e _ i r r a p 2 1 r
_ e n m _ a k e n u , 8 d e k a g n e d _ g k _
. n k _ 0 1 . 2 1 _ l u p 0 e _ 0 3 . 5 1 _ l u
k u 2 h a m _ n a l a j _ e k _
```

Untuk memudahkan Mr.X mengirim pesan kepada atasannya, bantulah Mr.X dengan sebuah program yang dapat mengubah pesan yang dikirmkan ke dalam bentuk di atas.

[input]

Max. 160 karakter

```
Hari ini mereka mengirim barang dengan mobil
pick-up pukul 12.10 ke jalan nirmala no.22,
pukul 15.30 ke jalan asoka, dan pukul 18.02 ke
jalan mahendradata
```

[output]

```
_ _ _ _ _ a s o k a , _ d a n _ p _ _ n _ n i r
m a l a _ u a n a i l _ p i c k - n k t a l b i
r i m _ b u o u a l a o g n I _ m a p . 1 d a j
m n i H a e r _ 2 _ a j _ _ e _ i r r a p 2 1 r
_ e n m _ a k e n u , 8 d e k a g n e d _ g k _
. n k _ 0 1 . 2 1 _ l u p 0 e _ 0 3 . 5 1 _ l u
k u 2 h a m _ n a l a j _ e k _
```

Langkah-langkah pembuatan program :

1. Inisialisasi panjang pesan maximal yang diinputkan user

```
#define MAX_PESAN 160
```

2. Inisialisasi array dan pointer penampung pesan serta matrix yang digunakan untuk meng-enkripsi pesan

```
char matrix[13][13];
char pesan[MAX_PESAN];
char *ppesan;
```

3. Inisialisasi variabel global yang digunakan dalam proses penyandian pesan

```
int jum_pesan,jum_matrix,baris,kolom = 0;
```

4. Prototype fungsi pengisian matrix

```
void isi_matrix(int batas, char arah[5]);
```

5. Inisialisasi variabel lokal untuk proses penyandian pesan

```
int i,j,batas_kanan,batas_kiri,batas_atas,batas_bawah;
```

7. Inputan pesan

```
printf("Masukan Pesan yang akan disandikan : \n\n\t"); gets(pesan);

ppesan = &pesan;
```

8. Pengecekan jumlah karakter dari pesan yang diinputkan oleh user

```
jum_pesan = strlen(pesan);
if(jum_pesan > MAX_PESAN){
    printf("\n\nPesan yang dimasukan terlalu panjang!!, silakan ulangi lagi\n\n\n\n");
    main();
}
else if(jum_pesan == 0){
    printf("\n\nTidak ada pesan yang Anda masukan, silakan ulangi lagi!!\n\n\n\n");
    main();
}
```

9. Proses pengisian matrix

```
for(i=0; i<7; i++){

    if(i == 0){
        baris = kolom = 6;
        isi_matrix(0,"tengah");
    }
    else{
        batas_kanan = batas_bawah = 6 + i;
        batas_kiri = batas_atas = 6 - i;

        baris = 7 - i;
        kolom = batas_kanan;

        isi_matrix(batas_bawah,"bawah");
        isi_matrix(batas_kiri,"kiri");
        isi_matrix(batas_atas,"atas");
        isi_matrix(batas_kanan,"kanan");
    }
}
```

10. Mencetak isi matrix

```
//cetak matrix
printf("\n\n\n====================== Matrix Enkripsi ======================\n\n");
for(i=0; i<13; i++){
    printf("  ");
    for(j=0; j<13; j++){
        printf("%c    ",matrix[i][j]);
    }
    printf("\n\n");
}
printf("\n=============================================================\n\n");
```

11. Mencetak hasil penyandian pesan

```
//cetak hasil enkripsi
printf("\n\nHasil Pesan yang telah disandikan : \n");
for(i=0; i<13; i++){
    for(j=0; j<13; j++){
        if(matrix[i][j] != ' ') printf("%c ",matrix[i][j]);
    }
}
```

12. Fungsi pengisian matrix

```
void isi_matrix(int batas, char arah[5]){

    while(0 == 0){
        if(jum_matrix <= jum_pesan){

            if(*ppesan == ' ' || *ppesan == '\0') matrix[baris][kolom] = '_';
            else matrix[baris][kolom] = *ppesan;

            ppesan++;
        }
        else{
            if(jum_matrix < MAX_PESAN) matrix[baris][kolom] = '_';
            else matrix[baris][kolom] = ' ';

        }
        jum_matrix++;

        if (arah == "tengah") break;
        else if(arah == "bawah"){ baris++; if(baris > batas){baris--; kolom--; break;}}
        else if(arah == "kiri"){ kolom--; if(kolom < batas){baris--; kolom++; break;}}
        else if(arah == "atas"){ baris--; if(baris < batas){baris++; kolom++; break;}}
        else if(arah == "kanan"){ kolom++; if(kolom > batas)break;}
    }

}
```

13. Ilustrasi Program

```
Masukan Pesan yang akan disandikan :

        Hari ini mereka mengirim barang dengan mobil pick-up pukul 12.10 ke jala
n nirmala no.22, pukul 15.30 ke jalan asoka, dan pukul 18.02 ke jalan mahendrada
ta


====================== Matrix Enkripsi ======================

  _   _   _   _
  _   a   s   o   k   a   ,   _   d   a   n   _   p
  _   _   n   _   n   i   r   m   a   l   a   _   u
  a   n   a   i   l   _   p   i   c   k   -   n   k
  t   a   l   b   i   r   i   m   _   b   u   o   u
  a   l   a   o   g   n   i   _   m   a   p   .   l
  d   a   j   m   n   i   H   a   e   r   _   2   _
  a   j   _   _   e   _   i   r   r   a   p   2   1
  r   _   e   n   m   _   a   k   e   n   u   ,   8
  d   e   k   a   g   n   e   d   _   g   k   _   .
  n   k   _   0   1   .   2   1   _   l   u   p   0
  e   _   0   3   .   5   1   _   l   u   k   u   2
  h   a   m   _   n   a   l   a   j   _   e   k   _

================================================================


Hasil Pesan yang telah disandikan :
_ _ _ _ _ a s o k a , _ d a n _ p _ _ n _ n i r m a l a _ u a n a i l _ p i c k
- n k t a l b i r i m _ b u o u a l a o g n i _ m a p . l d a j m n i H a e r _
2 _ a j _ _ e _ i r r a p 2 1 r _ e n m _ a k e n u , 8 d e k a g n e d _ g k _
. n k _ 0 1 . 2 1 _ l u p 0 e _ 0 3 . 5 1 _ l u k u 2 h a m _ n a l a j _ e k _
```

7. Laporan

Buatlah Program untuk mengecek kata atau kalimat palindrome dengan menggunakan function

Laporan diketik rapih pada kertas A4 dan dikumpulkan pada saat demo program.

Format Penulisan Laporan :

- Font Tulisan : Times New Roman 12 pt
- Font Source Code : Courier New 10 pt
- Line Spacing : Single

- Margin : 4, 3, 3, 3 (left, top, right, bottom)
- Page Numbering : a). cover = tanpa halaman
  b). kata pengantar s/d sebelum BAB I = center bottom, angka romawi kecil (i,ii,iii,)
  c). BAB = center bottom, angka arab (1,2,3)
  d). bagian BAB = top right, angka arab (1,2,3)
- Isi Laporan :

  BAB I PENDAHULUAN
  (Latar Belakang, Tujuan, Manfaat)

  BAB II LANDASAN TEORI

  BAB III PEMBAHASAN

  (tugas pendahuluan dan tugas praktikum)

  BAB IV PENUTUP

  (kesimpulan dan saran)