

2023/2024(2)
EF234201 Data Structure
Lecture #2

Array: Searching

Misbakhul Munir **IRFAN SUBAKTI**

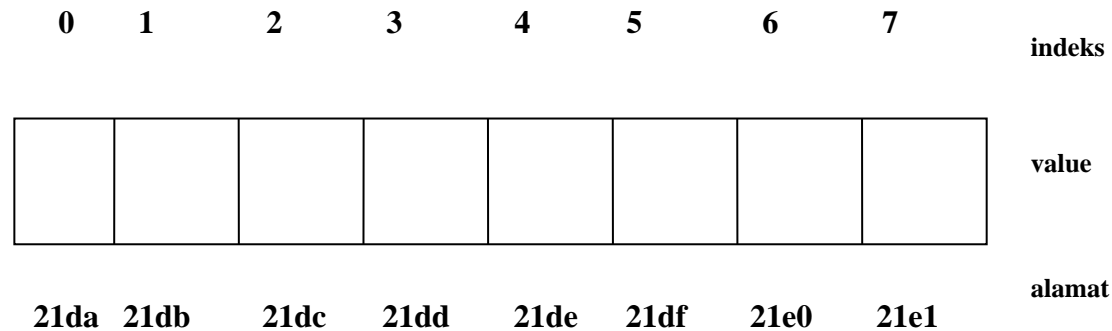
司馬伊凡

Мисбакхул Мунир **Ирфан Субакти**

Array: Definition

- Array: a finite ordered set of homogeneous elements
- Array elements are arranged in rows and can be accessed randomly in memory.
- Arrays have adjacent/adjacent addresses depending on the width of the data type.
- Arrays can be 1-dimensional, 2-dimensional, or even n -dimensional arrays.
- Array elements are of the same data type and can contain the same or different values.

char: 1-Dimensional Array Illustration

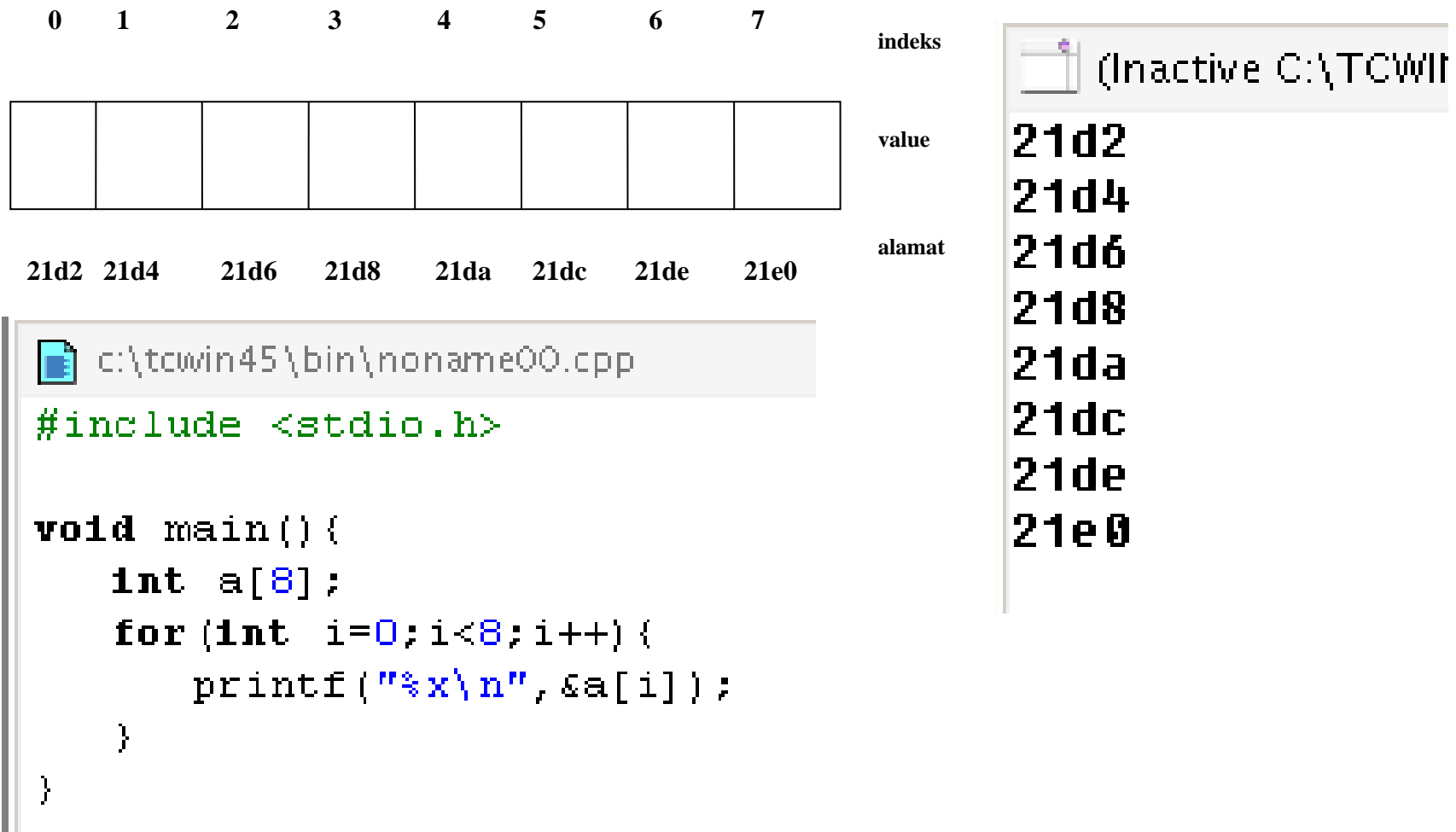


(Inactive C:\TC
21da
21db
21dc
21dd
21de
21df
21e0
21e1

```
#include <stdio.h>

void main() {
    char a[8];
    for (int i=0; i<8; i++) {
        printf("%x\n", &a[i]);
    }
}
```

int: 1-Dimensional Array Illustration



Array Elements: Accessing

- Array elements can be accessed by the program using a certain index randomly or sequentially
- Filling in and retrieving values for a particular index can be done by setting the value or displaying the value for the index in question.
- In C, there is no error handling regarding index value limits, whether the index is within a defined array index or not. This is the programmer's responsibility. So if the programmer accesses the wrong index, the resulting value will be different or damaged because it accesses an inappropriate memory address.

1-dimensional array: Example

```
char letter[9];
```

```
int age[10];
```

```
int condition[2] = {0,1}
```

```
int arr_dynamic[] = {1,2,3}
```

- The [] sign is also called the "th element...". For example, condition[0] means the zeroth element of the condition array.
- Arrays that have been ordered, for example, 10 places do not have to be filled in, they can only be filled with 5 elements, either sequentially or not. However, in the condition that it is not filled, there are still 10 booking places in memory, so places that are not filled will still be booked and left empty.
- We cannot declare a dynamic array without initialisation!

Other examples

- How to input and display arrays?
- Manipulation of 1-dimensional arrays?
- Arrays without initialisation are displayed immediately?
- Array initialised with 0?
- Array initialisation only the first 2 elements?

Array: Input-output

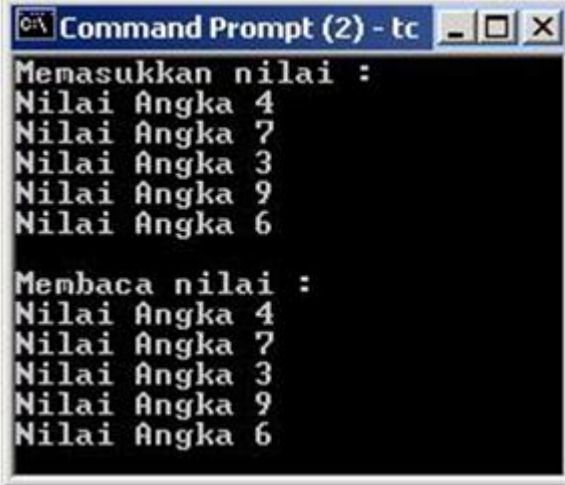
```
#include <stdio.h>
#include <conio.h>

void main()
{ int nilai[5], x;
  clrscr();

  printf("Memasukkan nilai :\n");
  for(x=0;x<5;x++)
  {
    printf("Nilai Angka : "); scanf("%d",&nilai[x]);
  }
  printf("\n");

  printf("Membaca nilai :\n");
  for(x=0;x<5;x++)
  {
    printf("Nilai Angka : %d",nilai[x]);
  }

  getch();
}
```



```
Command Prompt (2) - tc
Memasukkan nilai :
Nilai Angka 4
Nilai Angka 7
Nilai Angka 3
Nilai Angka 9
Nilai Angka 6

Membaca nilai :
Nilai Angka 4
Nilai Angka 7
Nilai Angka 3
Nilai Angka 9
Nilai Angka 6
```


Array: Manipulation

```
#include <stdio.h>
#include <conio.h>

int main(){
    int bil[7],i;
    printf("elemen 1 ? ");scanf("%d",&bil[0]);
    bil[1] = 5;
    bil[2] = bil[1] + 20;
    for(i=4;i<7;i++) bil[i] = i*10;
    bil[3] = bil[bil[1]];
    for(i=0;i<7;i++) printf("bil[%d] = %d dan alamatnya: %x\n",i,bil[i],&bil[i]);
    getch();
    return 0;
}
```

C:\TCWIN45\BIN\NONAME00.EXE

```
elemen 1 ? 25
bil[0] = 25 dan alamatnya: 2384
bil[1] = 5 dan alamatnya: 2386
bil[2] = 25 dan alamatnya: 2388
bil[3] = 50 dan alamatnya: 238a
bil[4] = 40 dan alamatnya: 238c
bil[5] = 50 dan alamatnya: 238e
bil[6] = 60 dan alamatnya: 2390
```

Initialisation

```
#include <stdio.h>
#include <conio.h>

int main() {
    int bil[7]; //tanpa inisialisasi
    for (int i=0; i<7; i++) {
        printf("Elemen ke-%i = %d, \talamat: %x\n", i, bil[i], &bil[i]);
    }
    getch();
    return 0;
}
```

```
#include <stdio.h>
#include <conio.h>

int main(){
    int bil[7] = {0}; //inisialisasi 0
    for (int i=0; i<7; i++){
        printf("Elemen ke-%i = %d\talamat: %x\n", i, bil[i], &bil[i]);
    }
    getch();
    return 0;
}
```

```
C:\TCWIN45\BIN\NONAME00.EXE
Elemen ke-0 = -1,      alamat: 21f0
Elemen ke-1 = 3224,   alamat: 21f2
Elemen ke-2 = 3182,   alamat: 21f4
Elemen ke-3 = -1,    alamat: 21f6
Elemen ke-4 = 8653,   alamat: 21f8
Elemen ke-5 = 9095,   alamat: 21fa
Elemen ke-6 = 19201,  alamat: 21fc
```

```
C:\TCWIN45\BIN\NONAME00.EXE
Elemen ke-0 = 0 alamat: 21fc
Elemen ke-1 = 0 alamat: 21fe
Elemen ke-2 = 0 alamat: 2200
Elemen ke-3 = 0 alamat: 2202
Elemen ke-4 = 0 alamat: 2204
Elemen ke-5 = 0 alamat: 2206
Elemen ke-6 = 0 alamat: 2208
```

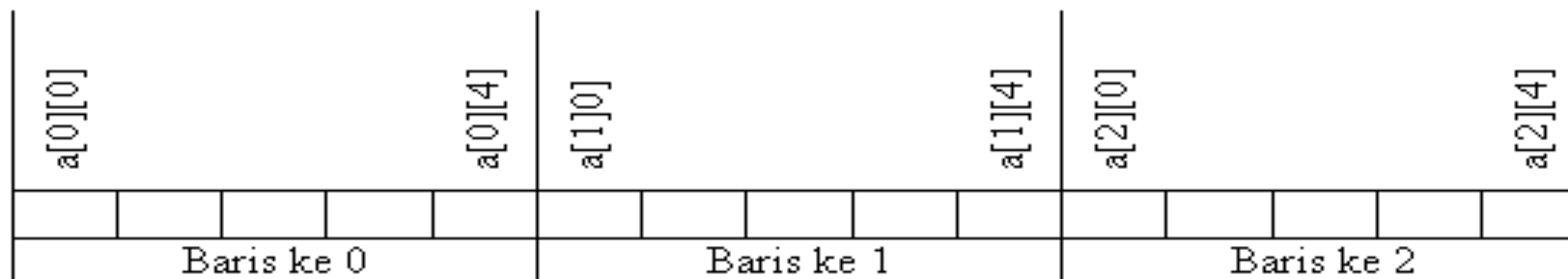
Array: Operations

- Addition of array elements
- Displays array elements
- Search for array elements
 - Search, if found, say FOUND!
- Removal of array elements
 - Search, if found then delete!
- Editing array elements
 - Search, if found then edit!

2-dimensional array

- `char a[3][5]`
- The same as a 3×5 matrix
- In reality (memory, RAM):

	0	1	2	3	4
0					
1					
2					



2-dimensional array: Illustration

```
#include <stdio.h>
#include <conio.h>

int main() {
    char a[3][5];
    for (int i=0; i<3; i++) {
        for (int j=0; j<5; j++) {
            printf("%x ", &a[i][j]);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

```
21d4 21d5 21d6 21d7 21d8
21d9 21da 21db 21dc 21dd
21de 21df 21e0 21e1 21e2
```

Array: Addressing

- Question: int A[10], it's known that &A[0] = H1000
 - What is &A[7]?
- Answer:
 - int is 2 bytes in size
 - Displacement $7-0 = 7 * 2 \text{ bytes} = 14 \text{ bytes}$
 - Then $H1000 + 14 = H100E$
- Question: int A[3][5], &A[0][0] = H1000
 - What is &A[2][3]?
- Answer:
 - int 2 bytes
 - Row shift: $2-0 = 2 * 5 \text{ (its column)} = 10$
 - Column shift: $3-0 = 3$
 - Total displacement: $10 + 3 = 13 * 2 \text{ bytes} = 26 \text{ bytes}$
 - Then $H1000 + 26 = H101A$

	0	1	2	3	4
0					
1					
2					

Searching

- Data often requires re-reading information (information retrieval) by searching.
- Searching is searching for data by tracing the data.
- The data search location can be an array in memory, it can also be a file on external storage.

Sequential Search

- It is a data search technique in an array (1-dimension) that will search all the array elements from start to finish, where the data does not need to be sorted first.
- The best possibility (best case) is if the data sought is located at the leading array index (first array element) so that the time needed to search for data is very short (minimal).
- The worst case is if the data being searched for is located at the last array index (last array element) so that the time required to search for data is very long (maximum).

Sequential Search (continued)

- For example, there is a one-dimensional array as follows:

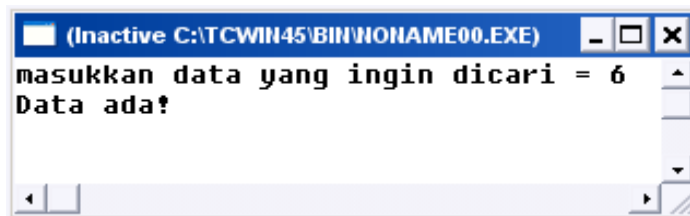
0	1	2	3	4	5	6	7	
								indeks
8	10	6	-2	11	7	1	100	value
21da	21db	21dc	21dd	21de	21df	21e0	21e1	alamat

- Then the program will ask for the data to be searched, for example, 6.
- If it is there, it will display the words "FOUND", whereas if it is not there it will display the words "NOT FOUND".

Sequential Search: Program

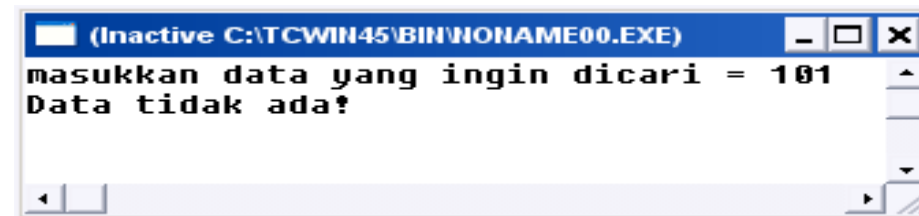
```
#include <stdio.h>
#include <conio.h>

void main(){
    clrscr();
    int data[8] = {8,10,6,-2,11,7,1,100};
    int cari;
    int flag=0;
    printf("masukkan data yang ingin dicari = ");    scanf("%d",&cari);
    for(int i=0;i<8;i++){
        if(data[i] == cari) flag=1;
    }
    if(flag==1) printf("Data ada!\n");
        else printf("Data tidak ada!\n");
}
```



(Inactive C:\TCWIN45\BIN\WONAME00.EXE)

```
masukkan data yang ingin dicari = 6
Data ada!
```



(Inactive C:\TCWIN45\BIN\WONAME00.EXE)

```
masukkan data yang ingin dicari = 101
Data tidak ada!
```

Program: Discussion

- The program uses a flag variable which is useful for indicating whether or not the data being searched for is present in the data array. Only valued 0 or 1.
- The flag is first initialised with the value 0.
- If found, the flag will be set to 1, if not, the flag will remain at 0.
- All elements of the data array will be compared one by one with the data searched for and entered by the user.
- Question: What if the data you are looking for is found and queried at what index?

Q & A

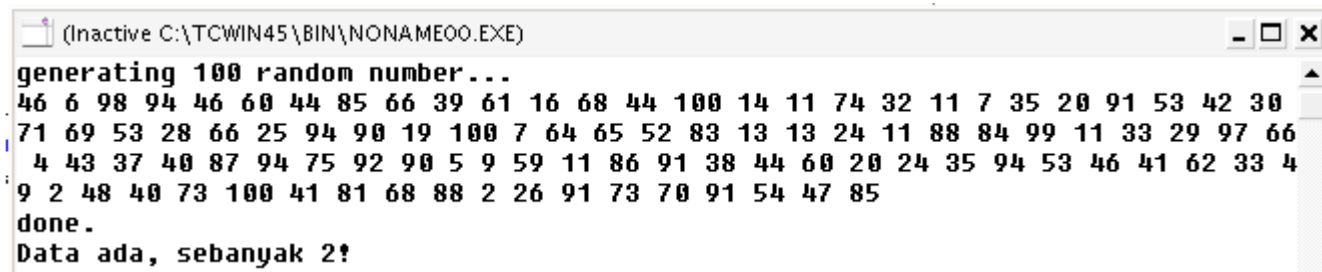
- Question: Is the method above efficient? If there are 10,000 data and all data is guaranteed to be unique?
- Solution: To increase efficiency, if the data you are looking for has been found, the loop must stop!
 - Hint: Use `break`!
- Question: How do I count how many data in an array are non-unique, whose value is the same as the data the user is looking for?
 - Hint: Use a counter variable with a value that will always increase if any data is found!

Example

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

void main() {
    clrscr();
    int data[100];
    int cari=20;
    int counter=0;
    int flag=0;
    randomize();
    printf("generating 100 random number...\n");
    for (int i=0;i<100;i++) {
        data[i]=random(100)+1;
        printf("%d ",data[i]);
    }
    printf("\ndone.\n");

    for (i=0;i<100;i++) {
        if (data[i]==cari) {
            counter++;
            flag=1;
        }
    }
    if (flag==1) printf("Data ada, sebanyak %d!\n",counter);
    else printf("Data tidak ada!\n");
}
```



```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
generating 100 random number...
46 6 98 94 46 60 44 85 66 39 61 16 68 44 100 14 11 74 32 11 7 35 20 91 53 42 30
71 69 53 28 66 25 94 90 19 100 7 64 65 52 83 13 13 24 11 88 84 99 11 33 29 97 66
4 43 37 40 87 94 75 92 90 5 9 59 11 86 91 38 44 60 20 24 35 94 53 46 41 62 33 4
9 2 48 40 73 100 41 81 68 88 2 26 91 73 70 91 54 47 85
done.
Data ada, sebanyak 2!
```

Sequential Search with Sentinel

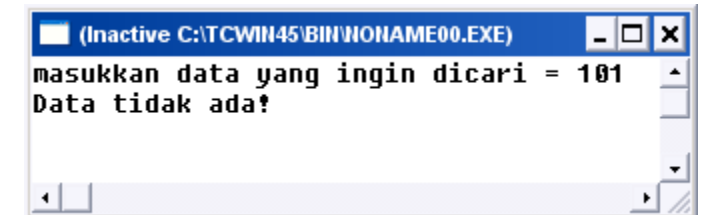
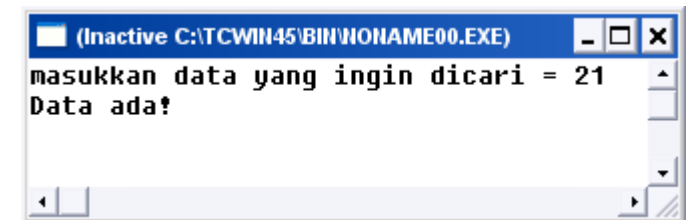
- Consider the following data array:

0	1	2	3	4	5	6	indeks
3	12	9	-4	21	6		value

- There are 6 pieces of data in the array (from index 0 to 5) and there is 1 additional array index (6th index) which does not yet contain data (called sentinel)
- The array at index 6 is useful for keeping data indexes at indexes 0 to 5 only. If the data search has reached the 6th array index, it means the data **DOES NOT EXIST**, whereas if the search does not reach the 6th index, then the data **EXISTS**.

Sequential Search with Sentinel: Program

```
#include <stdio.h>
#include <conio.h>
void main() {
    clrscr();
    int data[7] = {3, 12, 9, -4, 21, 6};
    int cari, i;
    printf("masukkan data yang ingin dicari = ");
    scanf("%d", &cari);
    data[6] = cari;
    i=0;
    while (data[i] != cari)
        i++;
    if (i<6)
        printf("Data ada!\n");
    else
        printf("Data tidak ada!\n");
}
```



Binary Search

- Existing data must first be sorted based on a certain sequence which is used as the search key.
- It is a deep data search technique by dividing the data into two parts each time the search process occurs.
- The principles of binary search are:
 - Data is taken from position 1 to end position N
 - Then find the middle data position with the formula: $(\text{start position} + \text{end position}) / 2$
 - Then the data sought is compared with the data in the middle, is it the same, smaller, or larger?
 - If it is greater, then the search process is searched with the starting position being the middle position + 1
 - If it is smaller, then the search process is searched with the final position being the middle position - 1
 - If the data is the same, it means they met.

Binary Search: Illustration

- Example: Data

- For example, the data you are looking for is 17

• 0	1	2	3	4	5	6	7	8
• 3	9	11	12	15	17	23	31	35
• A				B				C

- Because $17 > 15$ (middle data), then: beginning = middle + 1

• 0	1	2	3	4	5	6	7	8
• 3	9	11	12	15	17	23	31	35
•					A	B		C

- Because $17 < 23$ (middle data), then: end = middle - 1

• 0	1	2	3	4	5	6	7	8
• 3	9	11	12	15	17	23	31	35
•					A=B=C			

- Because $17 = 17$ (middle data), then it's FOUND!

Binary Search: Program

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void main(){
    clrscr();
    int data[9] = {3, 9, 11, 12, 15, 17, 23, 31, 35};
    int l, r, m;
    int n=9;
    int cari=17;
    l = 0;
    r = n-1;
    int ktm = 0;
    while (l<=r && ktm==0) {
        m = (l+r)/2;
        printf("data tengah: %d\n", m);
        if (data[m] == cari) ktm=1;
        else if (cari < data[m]) {
            printf("cari dikiri\n");
            r=m-1;
        }
        else {
            l=m+1;
            printf("cari di kanan\n");
        }
    }
    if (ktm==1) printf("data ada\n");
    else printf("data tidak ada\n");
}
```

```
data tengah: 4
cari di kanan
data tengah: 6
cari dikiri
data tengah: 5
data ada
```

Interpolation Search

- This technique is carried out on data that has been sorted based on certain keys
- This searching technique is carried out by estimating the location of the data.
 - Illustrative example: if we want to look for a name in a telephone book, for example, one that starts with the letter T, then we will not look for it from the beginning of the book, but we will immediately open it at 2/3 or 3/4 of the book's thickness.
- The formula for the relative position of the search key is calculated using the formula:

$$\text{position} = \frac{\text{key} - \text{data}[\text{low}]}{\text{data}[\text{high}] - \text{data}[\text{low}]} \times (\text{high} - \text{low}) + \text{low}$$

- If $\text{data}[\text{position}] > \text{data sought}$, $\text{high} = \text{pos} - 1$
- If $\text{data}[\text{position}] < \text{data sought}$, $\text{low} = \text{pos} + 1$

Interpolation Search: Case study

- For example, there is data as follows:

Kode	Judul Buku	Pengarang
025	The C++ Programming	James Wood
034	Mastering Delphi 6	Marcopolo
041	Professional C#	Simon Webe
056	Pure JavaScript v2	Michael Bolton
063	Advanced JSP & Servlet	David Dunn
072	Calculus Make it Easy	Gunner Christian
088	Visual Basic 2005 Express	Antonie
096	Artificial Life : Volume 1	Gloria Virginia

Case study: Solution

- Search Key? 088
- Low? 0
- High? 7
- Position = $(088 - 025) / (096 - 025) * (7 - 0) + 0 = [6]$
- Key[6] = search key, data found: Visual Basic 2005

- Search Key? 060
- Low? 0
- High? 7
- Position = $(060 - 025) / (096 - 025) * (7 - 0) + 0 = [3]$
- Key[3] < search key, then continue
- Low = $3 + 1 = 4$
- High = 7
- It turns out that Key[4] is 063 which is greater than 060.
- It means there is no 060 key.

Case study: Program

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

void main(){
    clrscr();
    int data[9] = {3,9,11,12,15,17,23,31,35};
    int low,high;
    int flag=0,d=36;
    float pos1;
    int pos,n=9;
    low=0;
    high=n-1;
    do{
        pos1 = (float) (d-data[low]) / (data[high]-data[low]) * (high-low) + low;
        pos = floor(pos1);
        if (data[pos] == d){
            flag=1;
            break;
        }
        if (data[pos] > d) high = pos-1;
        else
            if (data[pos] < d) low = pos + 1;
    } while (d >= data[low] && d <= data[high]);
    if(flag==0) printf("data tidak ada\n"); else printf("data ada\n");
}
```

Exercise

- Find out about how to use search websites (search engines) on the Internet and its technology!
 - Find out about Fibonacci Search!
-
- **NEXT: Sorting Arrays!**