INFORMATIKA UMN

UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

IF232
ALGORITHMS
&
DATA STRUCTURES

02

STRUCTURES, UNIONS, & ENUMERATIONS

DENNIS GUNAWAN

REVIEW

Arrays & Pointers:

Arrays

Pointers

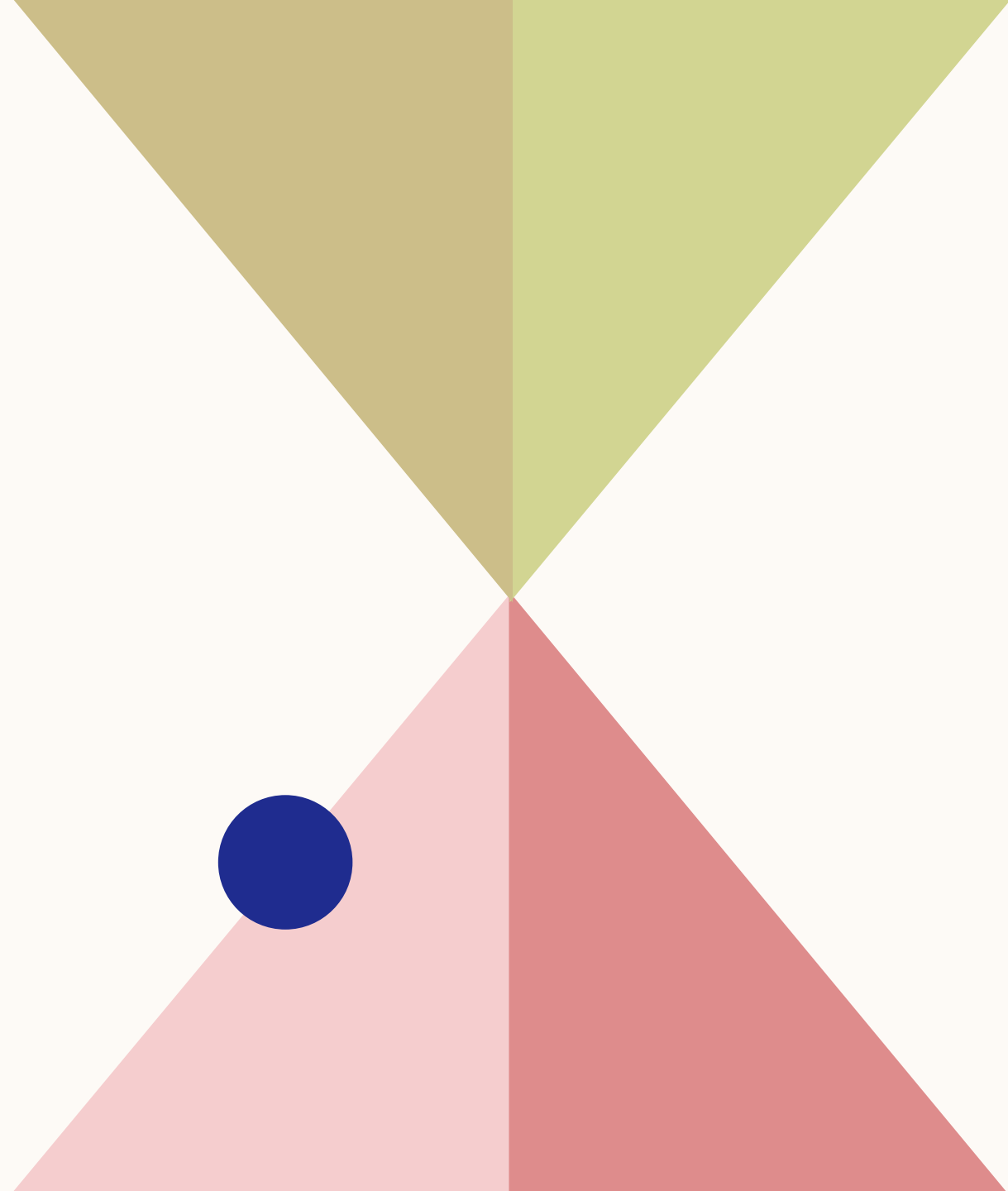
Strings

OUTLINE

Structures

Unions

Enumerations



STRUCTURES

- Collections of **related** variables under **one name**
- Structures may contain variables of many **different data types**
- Example: a **song** has **title**, **performer**, and **duration**

STRUCTURE DEFINITIONS

- Syntax

```
struct [structure_tag]
{
    data_type member_var_name [,member_var_name,...];
    [data_type member_var_name [,member_var_name,...];]
} [structure_var_name];
```

- Defining variables of structure types

```
struct structure_tag
```

```
struct struct_tag struct_var_name [,struct_var_name,...];
```

STRUCTURE DEFINITIONS

```
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

struct song track1;
```

```
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
}track1;
```

```
struct
{
    char title[50];
    char performer[35];
    int duration_ms;
}track1;
```

INITIALIZING STRUCTURES

- Syntax

```
struct struct_tag struct_var_name = {value1, value2, ..., valuen};
```

- Example

```
struct song track1 = {“Friend Like Me”, “Ne-Yo”, 182000};
```

ACCESSING STRUCTURE MEMBERS

- Structure member operator: dot operator (.)
  - The structure member operator accesses a structure member via **the structure variable name**

```
printf("%s\n",track1.title);
printf("%s\n",track1.performer);
printf("%d\n",track1.duration_ms);
```

ACCESSING STRUCTURE MEMBERS

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

int main()
{
    struct song track1 = {"Friend Like Me","Ne-Yo",182000};

    printf("Title          : %s\n",track1.title);
    printf("Performer      : %s\n",track1.performer);
    printf("Duration (ms)  : %d\n",track1.duration_ms);

    printf("\n[DG]");
    return 0;
}
```

```
Title          : Friend Like Me
Performer      : Ne-Yo
Duration (ms)  : 182000

[DG]
Process returned 0 (0x0)   execution time : 0.800 s
Press any key to continue.
```

ACCESSING STRUCTURE MEMBERS

- Structure pointer operator: arrow operator (->)
  - Consists of a minus (-) sign and a greater than (>) sign with no intervening spaces
  - The structure pointer operator accesses a structure member via **a pointer to the structure**

```
struct song *track1Ptr;

track1Ptr = &track1;

printf("%s\n",track1Ptr->title);
printf("%s\n",track1Ptr->performer);
printf("%d\n",track1Ptr->duration_ms);
```

```
(*track1Ptr).title
(*track1Ptr).performer
(*track1Ptr).duration_ms
```

?

ACCESSING STRUCTURE MEMBERS

```c
int main()
{
    struct song track1 = {"Friend Like Me","Ne-Yo",182000};
    struct song *track1Ptr;

    track1Ptr = &track1;

    printf("Title          : %s\n",track1Ptr->title);
    printf("Performer      : %s\n",track1Ptr->performer);
    printf("Duration (ms)  : %d\n",track1Ptr->duration_ms);

    printf("\n[DG]");
    return 0;
}
```

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};
```

```
Title          : Friend Like Me
Performer      : Ne-Yo
Duration (ms)  : 182000

[DG]
Process returned 0 (0x0)   execution time : 2.048 s
Press any key to continue.
```

GLOBAL SCOPE VS LOCAL SCOPE

```
[Local]  Song 1 : Friend Like Me
[Global] Song 2 : Part Of Your World

[DG]
Process returned 0 (0x0)   execution time : 1.017 s
Press any key to continue.
```

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

struct song track2 = {"Part Of Your World","Jessie J",200000};

int main()
{
    struct song track1 = {"Friend Like Me","Ne-Yo",182000};

    printf("[Local]  Song 1 : %s\n",track1.title);
    printf("[Global] Song 2 : %s\n",track2.title);

    printf("\n[DG]");
    return 0;
}
```

GLOBAL SCOPE VS LOCAL SCOPE

```c
int main()
{
    struct song
    {
        char title[50];
        char performer[35];
        int duration_ms;
    };

    struct song track1 = {"Friend Like Me","Ne-Yo",182000};

    printf("Title          : %s\n",track1.title);
    printf("Performer      : %s\n",track1.performer);
    printf("Duration (ms)  : %d\n",track1.duration_ms);

    printf("\n[DG]");
    return 0;
}
```

```
Title          : Friend Like Me
Performer      : Ne-Yo
Duration (ms)  : 182000

[DG]
Process returned 0 (0x0)   execution time : 0.977 s
Press any key to continue.
```

USING STRUCTURES WITH FUNCTIONS

- Structures may be passed to functions by passing individual structure members, by passing an entire structure, or by passing a pointer to a structure
- When structures or individual structure members are passed to a function, they are passed by value
- To pass a structure by reference (pointer), pass the address of the structure variable

USING STRUCTURES WITH FUNCTIONS

```c
#include <stdio.h>
#include <string.h>

struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

struct song newSong(char sTitle[], char sPerformer[], int sDuration)
{
    struct song s;

    strcpy(s.title,sTitle);
    strcpy(s.performer,sPerformer);
    s.duration_ms = sDuration;

    return s;
}
```

```c
int main()
{
    struct song track1 = newSong("Friend Like Me","Ne-Yo",182000);

    printf("Title          : %s\n",track1.title);
    printf("Performer      : %s\n",track1.performer);
    printf("Duration (ms)  : %d\n",track1.duration_ms);

    printf("\n[DG]");
    return 0;
}
```

```
Title          : Friend Like Me
Performer      : Ne-Yo
Duration (ms)  : 182000

[DG]
Process returned 0 (0x0)   execution time : 0.980 s
Press any key to continue.
```

USING STRUCTURES WITH FUNCTIONS

```c
#include <stdio.h>

struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

void printSong(struct song s)
{
    printf("Title          : %s\n",s.title);
    printf("Performer      : %s\n",s.performer);
    printf("Duration (ms)  : %d\n",s.duration_ms);
}
```

```c
int main()
{
    struct song track1 = {"Friend Like Me","Ne-Yo",182000};

    printSong(track1);

    printf("\n[DG]");
    return 0;
}
```

```
Title          : Friend Like Me
Performer      : Ne-Yo
Duration (ms)  : 182000

[DG]
Process returned 0 (0x0)   execution time : 0.373 s
Press any key to continue.
```

USING STRUCTURES WITH FUNCTIONS

```c
#include <stdio.h>
#include <string.h>

struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

void newSong(struct song *s, char sTitle[], char sPerformer[], int sDuration)
{
    strcpy((*s).title,sTitle);
    strcpy(s->performer,sPerformer);
    (*s).duration_ms = sDuration;
}
```

```c
int main()
{
    struct song track1;

    newSong(&track1,"Friend Like Me","Ne-Yo",182000);

    printf("Title          : %s\n",track1.title);
    printf("Performer      : %s\n",track1.performer);
    printf("Duration (ms)  : %d\n",track1.duration_ms);

    printf("\n[DG]");
    return 0;
}
```

```
Title          : Friend Like Me
Performer      : Ne-Yo
Duration (ms)  : 182000

[DG]
Process returned 0 (0x0)   execution time : 0.918 s
Press any key to continue.
```

ARRAYS OF STRUCTURES

```c
int main()
{
    int i;
    struct song track[3] = {{"Friend Like Me","Ne-Yo",182000},
                            {"Part Of Your World","Jessie J",200000},
                            {"Zero To Hero","Ariana Grande",159000}};

    for(i = 0;i < 3;i++){
        printf("%-18s\t%-13s\t%d ms\n",track[i].title,
                                       track[i].performer,
                                       track[i].duration_ms);

    }

    printf("\n[DG]");
    return 0;
}
```

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};
```

```
Friend Like Me          Ne-Yo           182000 ms
Part Of Your World      Jessie J        200000 ms
Zero To Hero            Ariana Grande   159000 ms

[DG]
Process returned 0 (0x0)   execution time : 0.370 s
Press any key to continue.
```

ARRAYS OF STRUCTURES

```c
int main()
{
    int i;
    struct song track[3];

    for(i = 0;i < 3;i++){
        scanf("%[^#]#%[^#]#%d",track[i].title,
                               track[i].performer,
                               &track[i].duration_ms);

        fflush(stdin);
    }

    printf("\n");

    for(i = 0;i < 3;i++){
        printf("%-18s\t%-13s\t%d ms\n",track[i].title,
                                       track[i].performer,
                                       track[i].duration_ms);

    }

    printf("\n[DG]");
    return 0;
}
```

```
Friend Like Me#Ne-Yo#182000
Part Of Your World#Jessie J#200000
Zero To Hero#Ariana Grande#159000

Friend Like Me          Ne-Yo           182000 ms
Part Of Your World      Jessie J        200000 ms
Zero To Hero            Ariana Grande   159000 ms

[DG]
Process returned 0 (0x0)   execution time : 51.201 s
Press any key to continue.
```

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};
```

NESTED STRUCTURES

```c
int main()
{
    int i;
    struct album album1 = {"We Love Disney",16};

    for(i = 0;i < 3;i++){
        scanf("%[^#]#%[^#]#%d",album1.track[i].title,
                               album1.track[i].performer,
                               &album1.track[i].duration_ms);
        fflush(stdin);
    }

    printf("\n\n%s\n\n",album1.title);

    for(i = 0;i < 3;i++){
        printf("%02d %-18s\t%-13s\t%d ms\n",i + 1,
                                            album1.track[i].title,
                                            album1.track[i].performer,
                                            album1.track[i].duration_ms);

    }

    printf("\n[DG]");
    return 0;
}
```

```
Friend Like Me#Ne-Yo#182000
Part Of Your World#Jessie J#200000
Zero To Hero#Ariana Grande#159000


We Love Disney

01 Friend Like Me       Ne-Yo           182000 ms
02 Part Of Your World   Jessie J        200000 ms
03 Zero To Hero         Ariana Grande   159000 ms

[DG]
Process returned 0 (0x0)
Press any key to continue.
```

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

struct album
{
    char title[50];
    int numberOfSongs;
    struct song track[3];
};
```

TYPEDEF

- The keyword **typedef** provides a mechanism for creating synonyms (or aliases) for previously defined data types
- Names for structure types are often defined with **typedef** to create shorter type names
- C programmers often use **typedef** to define a structure type, so a structure tag is not required

TYPEDEF

```c
struct song
{
    char title[50];
    char performer[35];
    int duration_ms;
};

typedef struct song playlist;
```

```c
typedef struct
{
    char title[50];
    char performer[35];
    int duration_ms;
} playlist;
```

```c
int main()
{
    playlist track1 = {"Friend Like Me","Ne-Yo",182000};

    printf("Title          : %s\n",track1.title);
    printf("Performer      : %s\n",track1.performer);
    printf("Duration (ms)  : %d\n",track1.duration_ms);

    printf("\n[DG]");
    return 0;
}
```

UNIONS

- A derived data type with members that **share the same storage space**
- A union shares the space instead of wasting storage on variables that are not being used
- The number of bytes used to store a union must be at least enough to hold the largest member
- In most cases, unions contain two or more data types

UNIONS

- Syntax

```
union [union_tag]
{
    data_type member_var_name [,member_var_name,...];
    [data_type member_var_name [,member_var_name,...];]
} [union_var_name];
```

- Defining variables of union types

```
union union_tag union_var_name;
```

UNIONS

```c
#include <stdio.h>

union data
{
    short number;
    char c[2];
};

int main()
{
    union data d;

    d.number = 17479;
    printf("Number    : %d\n",d.number);
    printf("c[1]c[0]  : %c%c\n",d.c[1],d.c[0]);

    printf("\n[DG]");
    return 0;
}
```

union	d															
member 1	number															
member 2	c[1]								c[0]							
bit	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1

```
Number    : 17479
c[1]c[0]  : DG

[DG]
Process returned 0 (0x0)   execution time : 1.466 s
Press any key to continue.
```

ENUMERATIONS

- A set of **integer enumeration constants represented by identifiers**
- Values in an enumeration start with 0, unless specified otherwise, and incremented by 1

ENUMERATIONS

- Syntax

```
enum [enum_tag]
{
    member1, member2, ..., membern
} [enum_var_name];
```

- Defining variables of enumeration types

```
enum enum_tag enum_var_name;
```

ENUMERATIONS

- Examples

```
enum months
{
    JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
};
```

```
enum months
{
    JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
};
```

ENUMERATIONS

```c
#include <stdio.h>

enum months
{
    JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC
};

int main()
{
    enum months month;
    char *monthName[13] = {"",
                           "January", "February", "March", "April",
                           "May", "June", "July", "August",
                           "September", "October", "November", "December"};

    for(month = JAN;month <= DEC;month++){
        printf("Month %2d: %s\n",month,monthName[month]);
    }

    printf("\n[DG]");
    return 0;
}
```

```
Month  1: January
Month  2: February
Month  3: March
Month  4: April
Month  5: May
Month  6: June
Month  7: July
Month  8: August
Month  9: September
Month 10: October
Month 11: November
Month 12: December

[DG]
Process returned 0 (0x0)
Press any key to continue.
```

PRACTICE

EXERCISES

1. Find the error in each of the following program segments and correct the error.

   a.

```
struct person{
    char lastName[15];
    char firstName[15];
    int age;
}

person d;
```

EXERCISES

1. Find the error in each of the following program segments and correct the error.

   b.

```
struct card{
    char *face;
    char *suit;
}

struct card c, *cPtr, cards[13];

cPtr = &c;

printf("%s\n",*cPtr->face);
printf("%s\n",cards.face);
```

EXERCISES

1. Find the error in each of the following program segments and correct the error.

   c.

```
union values{
    char w;
    float x;
    double y;
};

union values v = {1.27};
```

EXERCISES

2. Given the following struct and variable definitions.

```
struct customer{
    char lastName[15];
    char firstName[15];
    int customerNumber;
    struct{
        char phoneNumber[11];
        char address[50];
        char city[15];
        char state[3];
        char zipCode[6];
    } personal;
} customerRecord, *customerPtr;

customerPtr = &customerRecord;
```

EXERCISES

2. Write an expression that accesses the struct members in each of the following parts.

    a. Member `lastName` of struct `customerRecord`
    b. Member `firstName` of the struct pointed to by `customerPtr`
    c. Member `phoneNumber` of member `personal` of struct `customerRecord`
    d. Member `zipCode` of member `personal` of the struct pointed to by `customerPtr`

EXERCISES

3. What is the value of X?

```c
#include <stdio.h>

int multiple(int num)
{
    int mask = 1, mult = 1, i;

    for(i = 1; i <= 10; ++i, mask <<= 1){
        if((num & mask) != 0){
            mult = 0;
            break;
        }
    }

    return mult;
}
```

```c
int main()
{
    int y;

    puts("Enter an integer between 1 and 32000: ");
    scanf("%d", &y);

    if(multiple(y)){
        printf("%d is a multiple of X\n",y);
    }
    else{
        printf("%d is not a multiple of X\n",y);
    }

    return 0;
}
```

EXERCISES

4. What does the following program do?

```c
#include <stdio.h>

int mystery(unsigned int bits)
{
    unsigned int mask = 1 << 31;
    unsigned int total = 0, i;

    for(i = 1; i <= 32; ++i, bits <<= 1){
        if((bits & mask) == mask){
            ++total;
        }
    }

    return !(total % 2) ? 1 : 0;
}
```

```c
int main()
{
    unsigned int x;

    puts("Enter an integer: ");
    scanf("%u",&x);

    printf("The result is %d\n",mystery(x));

    return 0;
}
```

EXERCISES

5. Define a **structure** type named **long_lat** that would be appropriate for storing longitude or latitude values. Include components named **degrees** (an **integer**), **minutes** (an **integer**), and **direction** (**one of the characters** 'N', 'S', 'E', or 'W').

   The following are a structure type to represent a geographic location and a variable of this hierarchical structure type.

```
typedef struct
{
    char place[20];
    long_lat longitude, latitude;
} location_t;

location_t resort;
```

EXERCISES

5. Given that the values shown have been stored in **resort**.

`.place`	Hawaii\0		
`.longitude`	158	0	W
`.latitude`	21	30	N

Complete the following table.

Reference	Data Type of Reference	Value
resort.latitude	long_lat	21 30 'N'
resort.place	…	…
resort.longitude.direction	…	…
…	…	30
resort.place[3]	…	…

LAB

EXERCISES

1. Write a program to read and display the information about all the students in a class. Use structures and functions.

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 1
```

```
ID          : 001
Name        : Ethan William
Assignment  : 83
Mid Exam    : 72
Final Exam  : 78
```

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 1
```

```
ID          : 002
Name        : Jacob Oliver
Assignment  : 91
Mid Exam    : 84
Final Exam  : 86
```

EXERCISES

1. Write a program to read and display the information about all the students in a class. Use structures and functions.

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 2
```

Final Grade:

A	85 – 100	C+	60 – 64.99
A-	80 – 84.99	C	55 – 59.99
B+	75 – 79.99	D	45 – 54.99
B	70 – 74.99	E	0 – 44.99
B-	65 – 69.99		

Final Score:
30% * Assignment +
30% * Mid Exam +
40% * Final Exam

ID	Name	Assignment	Mid Exam	Final Exam	Final Score	Final Grade
001	Ethan William	83	72	78	77.70	B+
002	Jacob Oliver	91	84	86	86.90	A

EXERCISES

1. Write a program to read and display the information about all the students in a class. Use structures and functions.

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 3
```

```
Assignment : 87.00
Mid Exam   : 78.00
Final Exam : 82.00
Final Score: 82.30
```

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 1
```

```
ID         : 003
Name       : Liam Mason
Assignment : 75
Mid Exam   : 67
Final Exam : 65
```

EXERCISES

1. Write a program to read and display the information about all the students in a class. Use structures and functions.

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 2
```

Final Grade:

A	85 – 100	C+	60 – 64.99
A-	80 – 84.99	C	55 – 59.99
B+	75 – 79.99	D	45 – 54.99
B	70 – 74.99	E	0 – 44.99
B-	65 – 69.99		

Final Score:
30% * Assignment +
30% * Mid Exam +
40% * Final Exam

ID	Name	Assignment	Mid Exam	Final Exam	Final Score	Final Grade
001	Ethan William	83	72	78	77.70	B+
002	Jacob Oliver	91	84	86	86.90	A
003	Liam Mason	75	67	65	68.60	B-

EXERCISES

1. Write a program to read and display the information about all the students in a class. Use structures and functions.

```
1. Add student
2. Display student details
3. Display grade average

0. Exit

Menu: 3
```

```
Assignment : 83.00
Mid Exam   : 74.33
Final Exam : 76.33
Final Score: 77.73
```

REFERENCES

- Deitel, P. and Harvey Deitel (2022), C How to Program (9th Edition), Pearson Education.
- Thareja, R. (2014), Data Structures Using C (2nd Edition), India: Oxford University Press.

NEXT

File Processing:

Data Hierarchy

Files and Streams

Sequential-Access File Processing

VISION

To become an **outstanding** undergraduate Computer Science program that produces **international-minded** graduates who are **competent** in software engineering and have **entrepreneurial spirit** and **noble character**.

MISSION

1. To conduct studies with the best technology and curriculum, supported by professional lecturer
2. To conduct research in Informatics to promote science and technology
3. To deliver science-and-technology-based society services to implement science and technology

Without hard work,
nothing grows but weeds.

UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

INFORMATIKA UMN

Have patience.
All things are difficult before they become easy.