Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Introduction to C

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Linked Lists

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Linked Lists

- The building blocks for linked lists are:
  1. Pointers: These are used to establish connections between nodes, enabling traversal and linking.
  2. Structs: These define the structure of each node, containing data and a pointer to the next node.
  3. Memory Management: Dynamic memory allocation is necessary for the flexibility of linked lists.
- We will cover these topics individually and then bring them together to construct a linked list.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structs

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

OBJECTIVE

- Understand the concept of structures in C programming.
- Learn how to define and declare structures.
- Explore the relationship between structures and functions.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Introduction to Structures

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Need for Structures in C Programming

- C programming often requires grouping related data together.
- Structures provide a way to create user-defined data types.
- Structures help organize and manage complex data efficiently.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Concept of a Structure

- A structure is a collection of different data types grouped together.
- It allows creating a custom data type with its own set of members.
- Members can be of any data type, including other structures.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Purpose of Structures

- Structures provide a way to represent real-world entities.
- They enable creating complex data structures for efficient data management.
- Structures facilitate passing multiple related pieces of data as a single unit.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Real-World Examples of Structures

- Storing information about a person (name, age, address, etc.).
- Representing a point in 2D or 3D space (x, y, z coordinates).
- Managing student records (roll number, name, grades, etc.).

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Advantages of Using Structures

- Structures enhance code readability and maintainability.
- They allow efficient data organization and access.
- Structures make it easier to work with complex data and relationships.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Defining and Declaring Structures

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Introduction - Structures in C

- Structures are user-defined data types in C that allow us to combine data items of different kinds.
- Structures are used to represent a record, which allows you to store different types of data.
- Example: If you want to store information about a student (like name, age, and GPA), you can use a structure.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Introduction - Importance of Structures

- Structures provide a method for packing together data of different types.
- A structure is a convenient tool for handling a group of logically related data items.
- Structures help in organizing complex data in a more meaningful way.
- Abstract Data Type = Student, point, truck, Audio
- Primitive Data Type = int, float, char, double

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structure Definition - Syntax for defining a structure:

- ‘struct’ keyword is used to define a structure.
- ‘structureName’ is the name of the structure.
- ‘datatype’ is the type of the member (like int, float, char, etc.).
- ‘member1’, ‘member2’, etc. are names of the structure's members.

```
struct structureName {
    //dataType member1;
    //dataType member2;
    //...
};
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structure Members

- A structure member is a variable that is defined inside a structure.
- Each member has its own data type.
- Members can be of any valid C data type, including basic types (like int, char, etc.), arrays, pointers, and even other structures.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structure Members
Defining Structure Members

- Structure members are defined within the structure definition.
- Each member definition is a normal variable definition, like int a; or char name[50];.
- Members are separated by semicolons.

```
struct Student {
    char name[50]; // Member 1: name
    int age;       // Member 2: age
    float gpa;     // Member 3: gpa
};
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Declaring Structure Variables

Syntax for declaring a structure variable:

- Structure variables are instances of the structure.
- structureName is the name of the structure and variableName is the name of the variable you want to declare.
- Structure variables consume the amount of memory equal to the sum of sizes of all members.

```
struct structureName variableName;
```

```
struct Student student1;
struct Student student2;
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Initializing Structures

- Initialization of structures is similar to initialization of arrays.
- Members of a structure can be initialized at the time of declaration.

```
struct structureName variableName = { "value1", "value2"};
variableName.member1;
```

```
struct Student student1 = { "John", 20, 3.5 };
struct Student student2 = { "Alice", 22, 3.8 };
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Accessing Structure Members

- Structure members can be accessed and modified using a structure variable.
- The dot operator (.) is used for accessing structure members.
- Syntax for accessing structure members:

```
struct structureName variableName;
variableName.member1;
```

```
struct Student student1 = { "John", 20, 3.5 };
struct Student student2 = { "Alice", 22, 3.8 };

printf("%s", student1.name); // Accessing structure member
student2.age = 23; // Modifying structure member
```

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structures and Functions

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structures and Functions

Relationship between Structures and Functions:

- Structures can be passed to functions as arguments just like basic data types.
- Functions can return structures.
- Structure members can be accessed within functions.
- Structure pointers can be used as function parameters.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Passing Structures as Function Arguments

How to pass structures as function arguments:

- Structures can be passed to functions either by value or by reference.
- Passing by value: A copy of the entire structure is passed to the function.
- Passing by reference: Only the memory address of the structure is passed, allowing the function to modify the original structure.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Passing Structures as Function Arguments

How to pass structures as function arguments:

- Structures can be passed to functions either by value or by reference.
- Passing by value: A copy of the entire structure is passed to the function.
- Passing by reference: Only the memory address of the structure is passed, allowing the function to modify the original structure.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Passing Structures as Function Arguments

Example:

```
struct Student student1 = { "John", 20, 3.5 };
struct Student student2 = { "Alice", 22, 3.8 };
printStudent(student1);
printStudent(student2);
```

```
void printStudent(struct Student s) {
    printf("Name: %s\n", s.name);
    printf("Age: %d\n", s.age);
    printf("GPA: %.2f\n", s.gpa);
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structure Pointers in Function Parameters

Use of structure pointers in function parameters:

- Using pointers is more efficient when dealing with large structures, because it avoids copying the entire structure.
- Structure pointers can be dereferenced using the -> operator.

```
struct Student student1 = { "John", 20, 3.5 };
struct Student student2 = { "Alice", 22, 3.8 };
printStudent(student1);
printStudent(student2);
updateGPA(&student1, 3.8);// Passing by reference
printStudent(student1);
```

```
void updateGPA(struct Student* s, float newGPA) {
    s->gpa = newGPA;
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Returning Structures from Functions

Returning structures from functions:

- Functions can also return structures.
- This allows functions to produce structured outputs.

```
struct Student createStudent(const char* name, int age, float gpa) {
    struct Student s;
    strcpy_s(s.name, name);
    s.age = age;
    s.gpa = gpa;
    return s;
}
```

```
struct Student student1 = createStudent("Alice", 22, 3.8);
printStudent(student1);
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Structure Scope and Lifetime

Concept of structure scope and lifetime:

- The scope of a structure type is the region of code where it can be used.
- The lifetime of a structure variable is the duration for which it exists in memory.
- Structures defined inside a function have a local scope and are automatically destroyed when the function ends.
- Structures defined outside of all functions have a global scope and exist for the lifetime of the program.
- Understanding scope and lifetime can help avoid bugs related to using a variable out of its scope or after its lifetime.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Recap and Practice Exercises

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Recap

Key Points:

- A structure is a user-defined data type in C that groups related variables of different data types.
- Structures can be passed to functions, returned from functions.
- Structure members can be accessed using the dot operator.
- Structure scope and lifetime depends on where the structure is defined.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Discussion

Research:

What are some real-world examples of structures?

How can structures improve the organization and readability of code?

How do structures compare with other data types in C, like arrays?

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

PRACTICE EXERCISES

Exercise 1: 3D Space Distance Calculator

Exercise 2: Student Age Calculator

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Data Types

Pointers

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

- In this session, we will review the important concepts of variables and memory, and explore their relationship to pointers.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

Recap Variables and Memory

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Review of Variables and Memory

- Variables serve as containers for storing data in a program.
- Each variable is assigned a memory location, which can be thought of as a unique address.
- The memory is organized into bytes, and each byte has a specific address.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Variables, Memory Addresses, and Data

- Variables are associated with memory addresses, which allow us to locate and manipulate the stored data.
- The memory address of a variable can be accessed using the address-of operator (&).
- By obtaining the memory address, we gain the ability to directly interact with the data stored in the variable.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Example

- Let's consider an example:

```
int num = 10;
```

- In this case, the variable num stores the value 10.
- Behind the scenes, num is assigned a memory location, which we can represent as an address, let's say 0x1000.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Visual Representation

- Here's a visual representation of the variable num in memory:

```
Variable: num
Memory Address: 0x1000
Value: 10
```

```
Address     |    Data
---------------------
0x0000      |    00101101
0x0001      |    11001010
0x0002      |    01010101
...         |    ...
...         |    ...
0xFFFF      |    11110000
```

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Recap

- To recap:
- Variables store data and have corresponding memory addresses.
- Memory addresses allow us to locate and manipulate the data stored in variables.
- Now that we have reviewed these concepts, we are ready to dive into the fascinating world of pointers!

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

Introduction to Pointers

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

What are Pointers?

- Pointers are variables that store memory addresses as their values.
- They play a crucial role in C programming, offering direct access to memory locations.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Memory Addresses

- Memory addresses are unique identifiers for each byte of memory in a computer.
- They provide a way to locate and access data stored in memory.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

How Pointers Work

- Pointers store memory addresses, allowing us to refer to the data stored at that address.
- They create a connection between variables and the memory they occupy.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Using Pointers

- Pointers enable efficient and flexible memory management in C.
- They allow for dynamic memory allocation, deallocation, and manipulation.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Storing and Accessing Data

- Pointers are used to store addresses of variables and data structures.
- By dereferencing pointers, we can access the data they point to.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Importance of Pointers

- Pointers are essential for tasks such as passing data by reference, working with arrays, and dynamic memory allocation.
- They provide a level of control and efficiency not possible with other variable types.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Summary

- Pointers are variables that store memory addresses.
- They provide direct access to memory and allow for flexible memory management.
- Pointers are crucial for tasks like passing data by reference, working with arrays, and dynamic memory allocation.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

Declaring and Initializing Pointers

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers Overview

- Pointers are variables that store memory addresses as their values.
- They allow us to access and manipulate data directly by pointing to its memory location.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Declaring Pointers

- Pointers are declared using the asterisk (*) notation in C.
- For example, to declare a pointer of type int, we write int* ptr;.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Example: Declaring and Initializing

- Let's see an example:

```
int num = 10;
int* ptr;       // Declaring a pointer of type int
int num = 10;   // Declaring an integer variable
ptr = &num;     // Initializing the pointer with the address of num
```

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Dereferencing Pointers

- Once a pointer is initialized, we can access the value it points to by dereferencing it.
- We use the dereference operator (*) to access the value.
- For example, to access the value stored in num using the pointer ptr, we write *ptr.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

```
printf("Pointers with Primative data types Integer:\n");
int myInt = 10;
int* myIntPointer = &myInt;
printf("Pointer Data (address of myInt): %p\n", myIntPointer);
printf("Pointer Address (address of myIntPointer): %p\n", &myIntPointer);
printf("myInt Pointer: %p\n", myIntPointer);
printf("Data in myInt: %d\n", myInt);
printf("Data in myIntPointer: %d\n\n\n", *myIntPointer);
```

```
---------Pointers Primative Data Types Integer---------
Pointer Data (address of myInt): 000000779EBEF804
Pointer Address (address of myIntPointer): 000000779EBEF828
myInt Pointer: 000000779EBEF804
Data in myInt: 10
Dereferenced myIntPointer: 10
```

```
printf("Pointers with Primative data types Double:\n");
double myDouble = 42.424242;
double* myDoublePointer = &myDouble;
printf("Pointer Data (address of myArray): %p\n", myDoublePointer);
printf("Pointer Address (address of pointer): %p\n", &myDoublePointer);
printf("myDouble Pointer: %p\n", myDoublePointer);
printf("Data in myDouble: %lf\n", myDouble);
printf("Dereferenced myDoublePointer: %lf\n\n\n", *myDoublePointer);
```

```
---------Pointers Primative Data Types Double---------
Pointer Data (address of myArray): 000000779EBEF848
Pointer Address (address of pointer): 000000779EBEF868
myDouble Pointer: 000000779EBEF848
Data in myDouble: 42.424242
Dereferenced myDoublePointer: 42.424242
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

```
int myArray[5] = {1,2,3,4,5};
int* pointer = myArray;
printf("---------Pointers with Arrays---------\n");
printf("Pointer (address of myArray): %p\n", pointer);
printf("Pointer Address (address of pointer): %p\n", &pointer);
printf("myArray Pointer: %p\n", myArray);
printf("Data in Array index '4': % d\n\n\n", myArray[4]);
```

```
---------Pointers with Arrays---------
Pointer (address of myArray): 000000779EBEF7B8
Pointer Address (address of pointer): 000000779EBEF7E8
myArray Pointer: 000000779EBEF7B8
Data in Array index '4':  5
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Summary

- Pointers are declared using the asterisk (*) notation in C.
- They can be initialized with the address of a variable using the address-of operator (&).
- Dereferencing pointers allows us to access the value stored at the memory address they point to.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Exercise

Instructions:

- Your goal is to write a function called AddByReference based on its behavior in the given code snippet.
- The PointerFunction takes an int parameter by reference, which means any changes made to the parameter inside the function will affect the original variable.
- Reverse engineer the behavior of PointerFunction to understand its purpose and functionality.
- Write the code for PointerFunction and test it using the provided test case in main.
- Ensure that the modified value of myIntToChange is correctly printed in the second printf statement in main.

Test Case:

- Input: myIntToChange = 1
- Expected Output:
- Before PointerFunction: 1
- After PointerFunction: 2
- Remember to focus on understanding the purpose of the code and implementing the required logic in the PointerFunction. Good luck with your coding exercise!

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointers

Pointer Arithmetic

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointer Arithmetic

- Pointer arithmetic involves performing calculations on pointers.
- It's particularly useful for manipulating arrays efficiently.
- Pointer arithmetic allows traversal of contiguous memory locations.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Accessing Values using Pointers

- Code Example: Accessing Value through a Pointer

```
int arr[] = { 1, 2, 3, 4, 5 };
int* ptr = arr; // Pointer points to the first element of the array

for (int i = 0; i < 5; i++) {
    printf("%d ", *ptr); // Accessing the element through the pointer
    ptr++; // Moving the pointer to the next memory location
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Pointer Arithmetic Operations

- Common pointer arithmetic operations include:
- Incrementing pointers: ptr++ moves the pointer to the next memory location.
- Decrementing pointers: ptr-- moves the pointer to the previous memory location.
- Addition and subtraction: ptr + n or ptr - n moves the pointer by n memory locations.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Summary

- Pointers allow us to access values through memory addresses.
- The dereference operator (*) retrieves the value pointed to by a pointer.
- Pointer arithmetic enables efficient array manipulation.
- It simplifies array traversal and element modification.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Closing

- In this section, we explored accessing values using pointers and introduced pointer arithmetic.
- We learned how to dereference pointers and discussed the relationship between pointers and variables.
- Additionally, we explored the importance of pointer arithmetic in array manipulation.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Exploring Pointers:
Arrays and Functions in C

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

The Interplay between Pointers and Arrays

- Arrays in C are closely related to pointers. An array's name acts as a pointer to its first element. For example, if we declare an array int arr[10];, arr is a pointer to arr[0].
- We can access array elements using pointers. For example, *(arr + i) is equivalent to arr[i]. Pointer arithmetic allows us to traverse an array using a pointer. For instance, incrementing a pointer arr++ moves the pointer to the next array element.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

The Interplay between Pointers and Arrays

- Dynamic memory allocation allows us to create arrays at runtime. We use malloc() or calloc() to allocate memory, which returns a pointer to the first byte of the allocated space. It's crucial to free this memory using free() when we're done to prevent memory leaks.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Leveraging Pointers to Access Array Elements

```c
int numbers[] = { 10, 20, 30, 40, 50 };
int* ptr = numbers; // Array decays into a pointer pointing to its initial element

for (int i = 0; i < 5; i++) {
    printf("Element %d: Value %d\n", i, *(ptr + i));
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

The Role of Pointers in Functions

- Pointers pave the way for achieving pass-by-reference functionality in C functions.
- By passing arguments via reference using pointers, it becomes possible to alter variables that exist outside the function's domain.
- This is especially beneficial when we want a function to modify the original variable, not just a temporary copy.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Utilizing Pointers in Functions for Variable Modification

```
#include <iostream>
void increment(int* numPtr) {
    (*numPtr)++; // Dereferencing the pointer to modify the actual variable
}

int main() {
    int num = 10;
    printf("Before increment: %d\n", num);

    increment(&num); // Transmit the address of 'num' to the function

    printf("After increment: %d\n", num);
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Understanding Pointers and Functions

- Pointers offer the ability to pass arguments by reference in C.
- Functions can alter variables outside their jurisdiction by employing pointers as parameters.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Conclusion

- In this unit, we delved into the intricate relationship between pointers and arrays. We discovered how arrays can decay into pointers, which simplifies element access. Moreover, we examined the use of pointers in functions to implement pass-by-reference behavior.

- Reflect on the ideas discussed in this session. The comprehension of pointers and their interaction with arrays and functions is a cornerstone of proficient C programming.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Struct Pointers

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Introduction to Struct Pointers in C

- A struct in C is a user-defined data type that can hold related data of various types.
- Struct pointers enable us to interact with these data structures in a dynamic way.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Accessing Struct Members via Pointers

- Struct members can be accessed directly using the arrow operator (->).
- Syntax: ptr->member;

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Dynamic Memory Allocation for Struct Pointers

- Memory for struct pointers can be dynamically allocated using functions like malloc or calloc.
- Syntax: ptr = (struct MyStruct*)malloc(sizeof(struct MyStruct));

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Practical Application of Struct Pointers

- Struct pointers are powerful tools for manipulating struct objects.
- Example:

```
struct Person {
    char name[50];
    int age;
};
int main()
{
    struct Person* personPtr;
    personPtr = (struct Person*)malloc(sizeof(struct Person));
    if (personPtr != NULL)
    {
        strcpy_s(personPtr->name, "John");
        personPtr->age = 25;
    }
    else
    {
        printf("Memory allocation failed!\n");
    }
```

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Memory Management in C

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

OBJECTIVE

- The aim of this lesson is to provide students with a strong understanding of memory management in C, including memory allocation, deallocation, and common issues such as memory leaks and dangling pointers.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Understanding the Essentials

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Introduction to Memory Management

- Definition of Memory Management: "In C, memory management involves manually handling the allocation and deallocation of memory during program execution."
- Importance of Memory Management in C: "Efficient memory management can prevent program crashes, enhance performance, and ensure secure execution."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

What is Memory Management?

- Memory Management explained: "Memory management is the programmer's responsibility in C. It involves requesting memory when needed (allocation) and freeing it for other uses when no longer required (deallocation)."
- The programmer's role: "Programmers manage memory using functions such as malloc(), calloc(), realloc(), and free(). Proper use of these functions can prevent memory leaks, crashes, and improve overall performance."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Importance of Memory Management

- Importance of efficient memory usage: "Efficient memory management ensures that programs run quickly and smoothly, avoiding the waste of system resources."
- Consequences of poor Memory Management: "Memory leaks can slow down or crash programs. Dangling pointers can cause undefined behavior. Uninitialized memory can lead to unpredictable results."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Impact of Improper Memory Management

- Memory Leaks: "When dynamically allocated memory isn't deallocated, it leads to memory leaks, which can slow down the program and eventually cause it to crash."
- Dangling Pointers: "A pointer pointing to deallocated memory is called a dangling pointer and can lead to undefined behavior if accessed."
- Uninitialized Memory: "Accessing uninitialized memory can also lead to unpredictable program behavior."

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Dynamic Memory Allocation

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Impact of Improper Memory Management

- Definition: "Dynamic Memory Allocation refers to the process of manually requesting memory during program execution."
- Importance: "Dynamic memory allocation is necessary when we do not know how much memory we need at compile time. Instead, we request and free memory at runtime as required."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Functions for Dynamic Memory Allocation

- malloc(): "The malloc() function allocates a specified number of bytes of memory and returns a pointer to the allocated memory."
- calloc(): "The calloc() function allocates memory for an array of a specified number of elements, initializes them to zero, and returns a pointer to the memory."
- realloc(): "The realloc() function changes the size of previously allocated memory and returns a pointer to the new memory."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Using Memory Allocation Functions

- Explanation and code examples showing how to use malloc(), calloc(), and realloc().
- Explanation: "These functions return a void pointer, which can be cast into a pointer of any form."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on: Practicing Memory Allocation

1. Define a Book struct with the following fields:
   - title: a string (you can use a character array) to hold the book's title.
   - author: a string to hold the author's name.
   - pages: an integer to hold the number of pages.
   - price: a float to hold the price of the book.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on: Practicing Memory Allocation

1. Define a Book struct with the following fields:
   - title: a string (you can use a character array) to hold the book's title.
   - author: a string to hold the author's name.
   - pages: an integer to hold the number of pages.
   - price: a float to hold the price of the book.

```
struct Book {
    char title[100];
    char author[50];
    int pages;
    float price;
};
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on: Practicing Memory Allocation

2. Write a function createBook that dynamically allocates memory for a Book struct and returns a pointer to it. This function should take the title, author, pages, and price as parameters and assign them to the appropriate fields in the struct.

```
struct Book* createBook(char* title, char* author, int pages, float price) {
    struct Book* bookPtr = (struct Book*)malloc(sizeof(struct Book));
    if (bookPtr == NULL) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    strcpy(bookPtr->title, title);
    strcpy(bookPtr->author, author);
    bookPtr->pages = pages;
    bookPtr->price = price;
    return bookPtr;
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on: Practicing Memory Allocation

3. Write a function displayBook that takes a Book struct pointer as a parameter and prints out its fields.

```
void displayBook(struct Book* bookPtr) {
    printf("Title: %s\n", bookPtr->title);
    printf("Author: %s\n", bookPtr->author);
    printf("Pages: %d\n", bookPtr->pages);
    printf("Price: %.2f\n", bookPtr->price);
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on: Practicing Memory Allocation

3. In your main function, use the createBook function to create a Book struct. Then use the displayBook function to print out its fields. Then Use the Free method to release it from memory.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Memory Deallocation

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Importance of Memory Deallocation

- Explanation: "Memory deallocation is the process of returning previously allocated memory back to the system."
- Importance: "It's important to deallocate memory to avoid memory leaks. Memory leaks occur when memory is no longer needed but hasn't been deallocated, leading to waste of memory resources."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on: Practicing Memory Allocation

- free(): "The free() function deallocates the memory previously allocated by malloc(), calloc(), or realloc()."

```
int* ptr = (int*)malloc(10 * sizeof(int));
// use ptr
free(ptr);
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Demonstration: Properly Deallocating Memory

- Explanation and code examples showing how to use free() to deallocate memory.
- Emphasis on ensuring that every allocation (malloc/calloc/realloc) has a corresponding deallocation (free).

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Common Errors with Memory Deallocation

- Forgetting to deallocate: "This leads to memory leaks, where allocated memory is never freed, leading to excessive memory usage."
- Deallocating too soon: "If you free memory that is still in use, subsequent use of that memory can lead to undefined behavior."
- Double freeing: "Freeing memory that has already been freed can lead to undefined behavior."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Hands-on Exercises

- Description of exercises that involve deallocating memory, modifying previous exercises to include memory deallocation. For example:
- "Modify the program from the previous exercise to free the array after it's no longer needed."
- "Create a program that creates a dynamic array, resizes it, and then frees it."
- Remind students to check for memory leaks by ensuring every allocation has a corresponding deallocation.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Memory Leaks

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Memory Leaks

- Definition: "Memory leaks occur when dynamically allocated memory is not freed after it is no longer needed."
- Consequences: "Memory leaks can lead to excessive memory usage, degraded performance, and even application crashes."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Dangling Pointers

- Definition: "A dangling pointer is a pointer that doesn’t point to a valid memory location. This usually happens when memory is freed while there are still pointers referencing it."
- Consequences: "Dereferencing a dangling pointer can lead to undefined behavior."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Avoiding Memory Leaks and Dangling Pointers

- Best Practices: "Always free memory after use, and set pointers to NULL after freeing memory to avoid dangling pointers."
- Tools: "Use tools like Valgrind or built-in debugger features to detect memory leaks."

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Practical Memory Management Strategies

- Best Practices: "Consistent use of memory management techniques, such as deallocating memory after use, avoiding excessive memory allocation, and using tools to detect memory leaks."
- Code Reviews and Testing: "Regular code reviews and rigorous testing can help identify and prevent memory issues early."

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Data Types
char pointer

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

STRINGS

- In C, const char* is a pointer to a constant character or a string literal. It is often used to represent read-only strings or string constants.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- Declaration and Initialization: When you declare a const char* variable, you are creating a pointer that can hold the memory address of a character or a string literal. For example:

```
const char* message; // Declaration of a pointer to a constant character
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- Declaration and Initialization: When you declare a const char* variable, you are creating a pointer that can hold the memory address of a character or a string literal. For example:

```
const char* message; // Declaration of a pointer to a constant character
```

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Data Types

Data Without defined Data

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Null

- NULL is a special value (often defined as zero) used to represent a null pointer in C programming. It indicates that the pointer does not currently point to any valid memory address.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Data Types
Strings

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- Declaration and Initialization: To declare a string variable, you need to define an array of characters with a specific size to hold the characters of the string. For example:

```
char greeting[6]; // Declaring a string with a size of 6 characters
```

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

STRINGS

- You can also initialize a string during declaration by assigning a sequence of characters within double quotation marks:

```
char greeting[] = "Hello"; // Declaring and initializing a string
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- Null Terminator: In C, strings are terminated with a null character '\0' (ASCII value 0). It marks the end of the string and allows functions to determine the length of the string. When initializing a string, the null terminator is automatically added by the compiler.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- Accessing String Characters: Individual characters in a string can be accessed using array indexing. The index starts from 0 for the first character. For example:

```
char greeting[] = "Hello";
char firstChar = greeting[0]; // Accessing the first character 'H'
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- String Functions: C provides several built-in string functions to perform operations on strings. Some commonly used functions are:

- strlen(): Returns the length of a string.
- strcpy(): Copies one string to another.
- strcat(): Concatenates two strings.
- strcmp(): Compares two strings.
- sprintf(): Writes formatted data to a string.
- strtok(): Breaks a string into tokens based on a delimiter.
- These functions are available in the <string.h> header file.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- Mutable and Immutable: In C, strings are immutable, which means you cannot change the contents of a string once it is defined. However, you can modify individual characters within the string.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Strings

- String Input: To read a string from the user, you can use the scanf_s() function with the %s format specifier. However, you need to ensure that the input does not exceed the size of the string array to avoid buffer overflow.

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

STRINGS

- It's important to note that C does not have a built-in string data type. Instead, it uses character arrays to represent strings. Handling strings in C requires careful memory management and adherence to proper string manipulation techniques to avoid errors and vulnerabilities.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Random Number Generator

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Random Number Generator

Random number generators are in almost all programming languages. Random number generators are not truly random but pseudo random. This means that even though a number appears random it has a mathematical formula that can be broken down to accurately predict the number being generated.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Random Number Generator

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    // Initialize random number generator with current time
    srand(time(0));

    // Generate a random number between 0 and 99
    int randomNumber = rand() % 100;

    printf("Random Number: %d\n", randomNumber);

    return 0;
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Random Number Generator

In the code above:

- We first include the necessary header files. stdlib.h is for the rand() and srand() functions, time.h is for the time() function, and stdio.h is for the printf() function.
- The srand(time(0)) line seeds the random number generator. It's important to seed the random number generator with a different value each time your program runs, otherwise rand() will generate the same sequence of numbers each time. time(0) returns the current time, which is a handy way to get a different seed each time.
- rand() % 100 generates a random number and uses the modulus operator to limit its range to 0-99.
- Finally, we print the random number to the console.