

**Submitted by:** Muhammad Raffey

**Submitted to:** Sir Ahmad Faraz

**Sap ID:** 70153209

**Department:** CS

**Section:** 2I

**Assignment no:** 3

**Structures and Pointers in C++**

**Intro to Structures**

**Definition, Syntax, and Declaration**

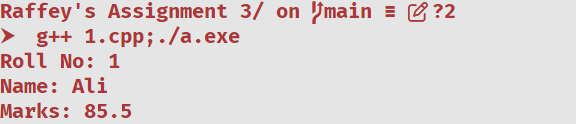
Structures in C++ are used to group variables of different types under one name. The `struct` keyword is used to define them.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;



struct Student

{

int rollNo;

string name;

float marks;

};

int main()

{

system("Color FC");

Student s1 = {1, "Ali", 85.5};

cout << "Roll No: " << s1.rollNo << endl;

cout << "Name: " << s1.name << endl;

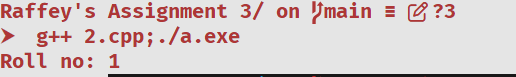
cout << "Marks: " << s1.marks << endl;

return 0;

}**Private/Public Members**

Structures can have private or public members. Public members can be accessed directly, while private members require methods.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

struct Student

{

private:

int rollNo;

public:

void setRollNo(int r) { rollNo = r; }

int getRollNo() { return rollNo; }

};

int main()

{

system("Color FC");

Student s1;

s1.setRollNo(1);

cout << "Roll no: " << s1.getRollNo() << endl;

// cout<<s1.rollNo; // Will Give error because rollNo is private

return 0;

}

**Modifying Members**

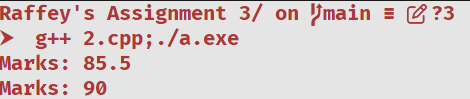
Members of a structure can be modified easily after initialization.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;



struct Student

{

int rollNo;

float marks;

};

int main()

{

system("Color FC");

Student s1 = {1, 85.5};

cout << "Marks: " << s1.marks << endl; // Output: 85.5

s1.marks = 90.0;

cout << "Marks: " << s1.marks << endl; // Output: 90

return 0;

}

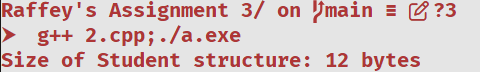
**Memory Allocation and Size**

The memory occupied by a structure depends on its members.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

struct Student

{

int rollNo;

char grade;

float marks;

};

int main()

{

system("Color FC");

Student student;

cout << "Size of Student structure: " << sizeof(Student) << " bytes" << endl;

return 0;}

**Arrays of Structures**

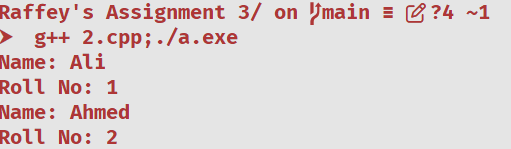
Structures can be used in arrays to store multiple objects.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;



struct Student

{

int rollNo;

string name;

};

int main()

{

system("Color FC");

Student students[2] = {{1, "Ali"}, {2, "Ahmed"}};

for (int i = 0; i < 2; i++)

{

cout << "Name: " << students[i].name << endl;

cout << "Roll No: " << students[i].rollNo << endl;

}

return 0;

}

**Nested Structures**

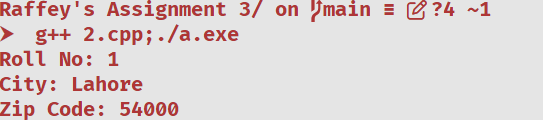
One structure can be a member of another structure.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

struct Address

{

string city;

int zipCode;

};

struct Student

{

int rollNo;

Address address;

};

int main()

{

system("Color FC");

Student s1 = {1, {"Lahore", 54000}};

cout << "Roll No: " << s1.rollNo << endl;

cout << "City: " << s1.address.city << endl;

cout << "Zip Code: " << s1.address.zipCode << endl;

return 0;

}

**Pointers to Structures**

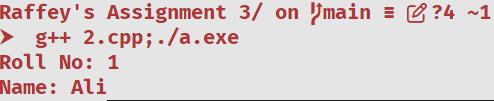
Pointers can be used to access structure members.

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

struct Student

{

int rollNo;

string name;

};

int main()

{

system("Color FC");

Student s1 = {1, "Ali"};

Student \*ptr = &s1;

cout << "Roll No: " << ptr->rollNo << endl;

cout << "Name: " << ptr->name << endl;

return 0;

}

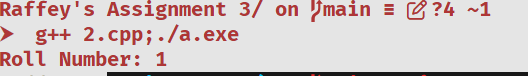
**Structures and Functions**

Structures can be passed to functions and returned as well.

#include <iostream>

#include <windows.h>

using namespace std;

struct Student

{

int rollNo;

};

void display(Student s)

{

cout << "Roll Number: " << s.rollNo;

}

int main()

{

system("Color FC");

Student s1 = {1};

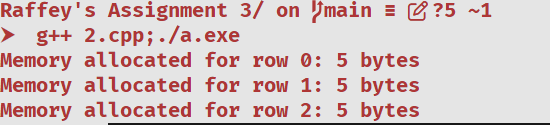
display(s1);

return 0;}

**2D Array with Pointers**

**Dynamic Memory Allocation**

A 2D array can be dynamically allocated using pointers to manage memory efficiently.

  
#include <iostream>

#include <windows.h>

using namespace std;

int main()

{

system("Color FC");

int rows = 3, cols = 5;

char \*\*array = (char \*\*)malloc(rows \* sizeof(char \*));

for (int i = 0; i < rows; i++)

{

array[i] = (char \*)malloc(cols \* sizeof(char));

cout << "Memory allocated for row " << i << ": " << cols \* sizeof(char) << " bytes" << endl;

}

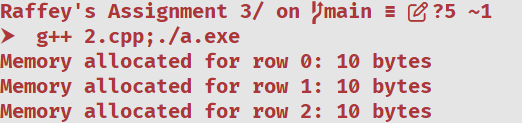
return 0;

}

**Operations on Character Arrays using Pointers**

Pointers can be used to perform operations on dynamically allocated character arrays.

#include <iostream>

#include <cstdlib>

#include <windows.h>

using namespace std;

int main()

{

system("Color FC");

int rows = 3;

char \*\*array = (char \*\*)malloc(rows \* sizeof(char \*));

for (int i = 0; i < rows; i++)

{

array[i] = (char \*)malloc(10 \* sizeof(char));

cout << "Memory allocated for row " << i << ": " << 10 \* sizeof(char) << " bytes" << endl;

}

return 0;

}