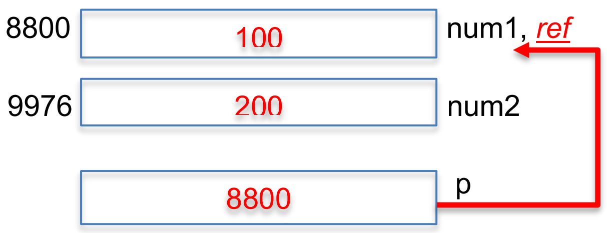
Tutorial 1 - Basic C++ Programming

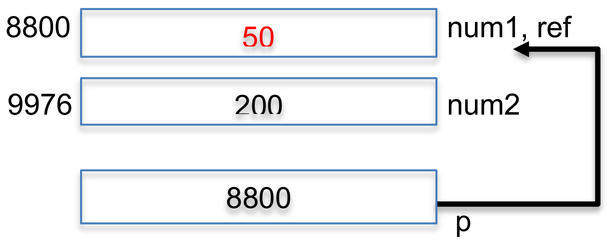
Q1.



Note: The code " int &ref = *p; " will make ref refers to num1, and is equivalent to " int &ref = num1; ". More specifically,

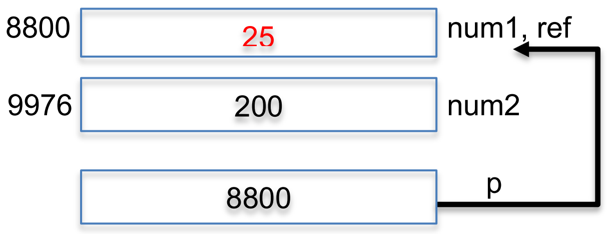
- *p dereferences p (i.e., access the value stored at the address that p is pointing to), which means *p refers to num1;
- ref is declared as a reference and is bound to *p, which is num1.
- Thus, ref is an alias of num1.

(i) *p = 50;



(a) num1 is changed to 50;
(b) num2 is still 200;
(c) p is pointing to the address of num1, i.e., 8800;
(d) *p is changed to 50;
(e)ref is still an alias of num1, so ref is 50;
(f) &ref is 8800, i.e., the address of num1.

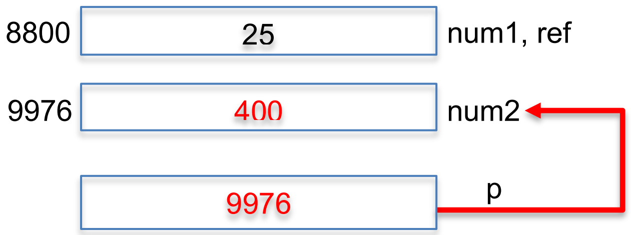
(ii) ref = ref / 2;



Note: ref is an alias of num1. So all the operations done on ref is executed to num1.

(a) num1 is changed to 25;
(b) num2 is still 200;
(c) p is still pointing to the address of num1, i.e., 8800;
(d) *p is changed to 25;
(e)ref is still an alias of num1, so ref is 25;
(f) &ref is 8800, i.e., the address of num1.

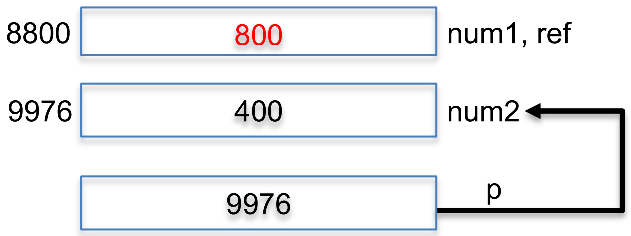
(iii) p = &num2; *p = 400;



Note: the change of p will not affect ref.

(a) num1 is still 25;
(b) num2 is changed to 400;
(c) p is pointing to the address of num2 now, i.e., 9976;
(d) *p is the value of num2, i.e., 400;
(e) ref is still an alias of num1 and is not changed, so ref is still 25;
(f) &ref is 8800, i.e., the address of num1.

(iv) ref = num2; ref = ref * 2;



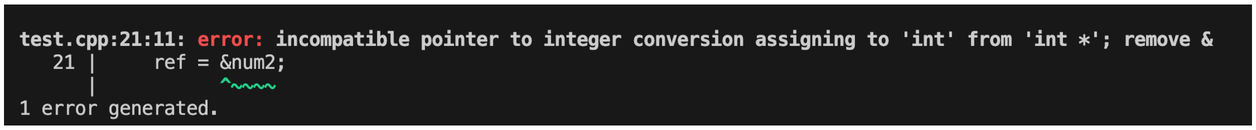
Note: the code "ref = num2;" does NOT mean that ref will become a reference to num2. Instead, ref is still an alias of num1, so the code "ref = num2;" is equivalent to the code "num1 = num2;".

(a) num1 is changed to be equal to the value of num2, i.e., 400; Then, it is updated to twice of its original value, i.e., 800;
(b) num2 is still 400;
(c) p is still pointing to the address of num2, i.e., 9976;

(d) *p is the value of num2, i.e., 400;
(e) ref is still an alias of num1 and is not changed, so ref is 800 now;
(f) &ref is 8800, i.e., the address of num1.

(v) ref = &num2;

Note: This line of code will result in a compiler error! ref is an alias of num1, i.e., an integer – “int”, while &num2 means the address of num2 – “int *”. They are different data types.



```
test.cpp:21:11: error: incompatible pointer to integer conversion assigning to 'int' from 'int *'; remove &
   21 |     ref = &num2;
      |           ^~~~~
1 error generated.
```

Q2.

```
#include <iostream>
using namespace std;

// Function to calculate area of a square
int calArea(int side) {
    return side * side;
}

// Function to calculate area of a rectangle
int calArea(int length, int width) {
    return length * width;
}

// Function to calculate area of a trapezoid
double calArea(int base1, int base2, int height) {
    return 0.5 * (base1 + base2) * height;
}


int main() {
    int choice;

    while (true) {
        // Display menu options
        cout << "\nChoose an option:\n";
        cout << "1 - Square\n";
        cout << "2 - Rectangle\n";
        cout << "3 - Trapezoid\n";
        cout << "Other - Exit\n";
        cout << "Enter your choice (int): ";
```

```cpp
        cin >> choice;

        if (choice == 1) {
            // Square
            int side;
            cout << "Enter the side length of the square (int): ";
            cin >> side;
            cout << "Area of Square: " << calArea(side) << endl;
        }
        else if (choice == 2) {
            // Rectangle
            int length, width;
            cout << "Enter the length and width of the rectangle (int): ";
            cin >> length >> width;
            cout << "Area of Rectangle: " << calArea(length, width) << endl;
        }
        else if (choice == 3) {
            // Trapezoid
            int base1, base2, height;
            cout << "Enter the two bases and height of the trapezoid (int): ";
            cin >> base1 >> base2 >> height;
            cout << "Area of Trapezoid: " << calArea(base1, base2, height) << endl;
        }
        else {
            // Exit program
            cout << "Exiting program..." << endl;
            break;
        }
    }

    return 0;
}
```

Q3.

Sample Solution

```cpp
#include <iostream>
using namespace std;

// Template function to calculate the area of a square
template <typename T>
T calArea(T side) {
    return side * side;
}

// Template function to calculate the area of a rectangle
template <typename T>
T calArea(T length, T width) {
    return length * width;
}

// Template function to calculate the area of a trapezoid
template <typename T>
T calArea(T base1, T base2, T height) {
    return (base1 + base2) * height / 2;
}

int main() {
    // Test cases
    int side1 = 5;
    cout << "Area of Square: " << calArea(side1) << endl;
    double side2 = 11.11;
    cout << "Area of Square: " << calArea(side2) << endl;

    int length1 = 10, width1 = 20;
    cout << "Area of Rectangle: " << calArea(length1, width1) << endl;
    float length2 = 23.4, width2 = 10.8;
    cout << "Area of Rectangle: " << calArea(length2, width2) << endl;

    long b1 = 20, b2 = 40, height = 10;
    cout << "Area of Trapezoid: " << calArea(b1, b2, height) << endl;

    return 0;
}
```

Q4.

Sample Solution

```
#include <iostream>
using namespace std;
union Result {
    int mark;
    char grade; // Can be only 'A', 'B' or 'C'
};
struct Student {
    char studentName[50];
    bool isGrade;
    int finalMark; // Used to store the final mark
    Result res;

    void convertGrade() {
        if (isGrade) { //The function can directly access the member variables
           switch (res.grade) {
                case 'A': finalMark = 90; break;
                case 'B': finalMark = 80; break;
                case 'C': finalMark = 60; break;
                default: finalMark = 0; break;
```

```
            }
        } else {
            finalMark = res.mark;
        }
    }
};
void displayStudentInfo(Student *students, int count);
int main() {
    int numStudents;
    cout << "How many students do you want to input?" << endl;
    cout <<"Enter student size: ";
    cin >> numStudents;
    cin.get(); // To clear the newline character

    Student *students = new Student[numStudents]; // Dynamic memory

    for (int i = 0; i < numStudents; i++) {
        cout << "Enter student name: ";
        cin.getline(students[i].studentName, 50); // Read the whole line

        char resultType;
        cout << "Enter 'G' if result is grade or 'M' if result is mark: ";
        cin >> resultType;

        if (resultType == 'G' || resultType == 'g') {
            students[i].isGrade = true;
            cout << "Enter grade (A,B,C): ";
            cin >> students[i].res.grade;
        } else {
            students[i].isGrade = false;
            cout << "Enter mark (0-100): ";
            cin >> students[i].res.mark;
        }
        cin.get(); // To clear the newline character

        students[i].convertGrade(); //Convert the grade or mark to finalMark
    }

    displayStudentInfo(students, numStudents);
    delete[] students; // Free allocated memory
    students = nullptr; //Prevent dangling pointer
    return 0;
}
void displayStudentInfo(Student *students, int count) {
    int totalMarks = 0;
    cout << "\nStudent Results:" << endl;
    for (int i = 0; i < count; i++) {
        cout << "Name: " << students[i].studentName << ", Final Mark: " <<
students[i].finalMark << endl;
        totalMarks += students[i].finalMark;
    }
    float average = (float) totalMarks / count; //Type conversion
    cout << "\nAverage Final Mark: " << average << endl;
}
```