Simple code output

#include<iostream>

int main()

{

    std::cout << "I like pizza!" << std::endl;

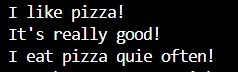
    std::cout << "It's really good!" <<'\n'; //<< std::endl; er poriborte <<'\n'; use kora jabe

    std::cout << "I eat pizza quie often!" << std::endl;

    return 0;

}

Result:



Variable sum code:

#include<iostream>

int main()

{

    int x; //declaration

    x = 5;

    int y = 6;

    int sum = x + y;

    std::cout << "x = " << x << '\n';

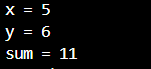
    std::cout << "y = " << y << '\n';

    std::cout << "sum = " << sum << '\n';

    return 0;

}

Result:



Datatypes code:

#include<iostream>

int main()

{

    int age = 21;

    int days = 7.5;

    std::cout << '\n';//eta diye faka line output niewa jabe

    std::cout << age << '\n';

    std::cout << days << '\n';/\*integer e 7.5 er moto decimal

                              with fractional part thakle output

                              e sudhu decimal point er ager purno

                            shonkha show korbe\*/

    std::cout << '\n';

    //double(number including decimal)

    double price = 10.99;

    double gpa = 2.5;

    std::cout << price << '\n';

    std::cout << gpa << '\n';

    std::cout << '\n';

    //single character

    char grade = 'A';

    char initial = 'BC'; /\* char datatype sudhu ekta character

                          store korte pare, ek er cheye beshi

                          character dile last character ta outpuut

                          e show korbe\*/

    char currency = '$'; //char datatype $,%,&,^ eulao store kore out dei

    std::cout << grade << '\n';

    std::cout << initial << '\n';

    std::cout << currency << '\n';

    std::cout << '\n';

    //boolean (true or false)

    bool student = false; //false hoile output 0

    bool power = true;    //true hoile output 1

    std::cout << student << '\n';

    std::cout << power << '\n';

    std::cout << '\n';

    /\*string(objects that represent a sequence of text)

    strings are provided from the standard namespace, to decalre

    a string we type std::string variablename = "name";  \*/

    std::string name = "Rangan";

    std::string team = "Argentina";

    std::string club = "Barcelona visca barca 5";//space and number soho sentence o output dite pare

    std::cout << name << '\n';

    std::cout << team << '\n';

    std::cout << club << '\n';

    std::cout << "Hello World" << '\n';//ebhabe directly string output newa jai

    std::cout << "Vamos " << team << '\n';/\*ebhabe directly string output newar por

                                          variable name diye directly lekha string er

                                          por oi variable output pawa jai

                                          and pay atention to spacing,

                                          ebhabe output newar shomoy  \*/

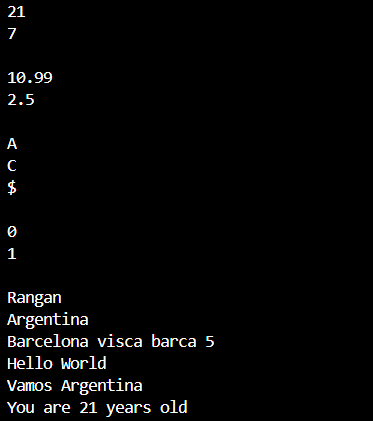
    std::cout << "You are " << age << " years old" << '\n';//ebhabeo output newa jabe

    std::cout << '\n';

    return 0;

}

Result:



Const keyword code:

#include<iostream>

int main()

{

    //The const keyword specifies that a variable's value is constant

    //tells the compiler to prevent anything from modifying it

    //(read-only)

    const double PI = 3.1416;

    const int LIGHT\_SPEED = 299792458;

    const int WIDTH = 1920;

    const int HEIGHT = 1080;

    double radius = 8.5;

    double circumference = 2 \* PI \* radius;

    std::cout << "Circumference = " << circumference << '\n';

    return 0;

}

Result:



Namespace code:

#include<iostream>

namespace first

{

    int x = 1;

}

namespace second

{

    int x = 2;

}

int main()

{

    /\*Namespace = provides a solutionn for preventing name conflicts

    in large projects. Each entity needs a unique name.

    A namespace allows for identically named entities

    as long as the amespaces are different.  \*/

    int x = 0;

    std::cout << "local version of x = " << x << '\n';

                                                /\*specically jodi kon

                                                x er value bole dewa na

                                                hoi taile local version

                                                output dibe\*/

    std::cout << "Version of x within the first namespace = " << first::x << '\n';

                                                /\*namespace er kono variable er

                                                output newar jonno namespace er

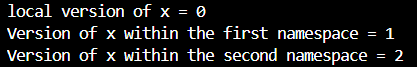
                                                name::variable\_name nite hobe\*/

    std::cout << "Version of x within the second namespace = " << second::x << '\n';

    return 0;

}

Result:



Using namespace code:

#include<iostream>

namespace first

{

    int x = 1;

}

namespace second

{

    int x = 2;

}

int main()

{

    using namespace first;

    std::cout << "x = " << x;

    /\*emon using namespace e jei namespace er kotha bola hobe

    output e shei output e dekhabe(ekhane <<first::x lekhar dorkar

    nei,jodi int\_main er moddhe oi variable declare kora na thake)\*/

    /\*but jodi int\_main er moddhe variable declare kora thake taile

    first namespace er output nite hole <<first::x; likhte hobe, jodi

    << x; emon lekha hoi taile int\_main er moddhe je variable decclare

    kora hoiche oita output dibe\*/

    /\* int-main e multiple using-namespace dewa jabena\*/

    /\*using namespace first diye <<second::x; likhle output ashbe seccond

    namespace er \*/

    return 0;

}

Result:



Using namespace std code:

#include<iostream>

using namespace std;

int main()

{

    string name = "Bro";

    cout << "Hello " << name;

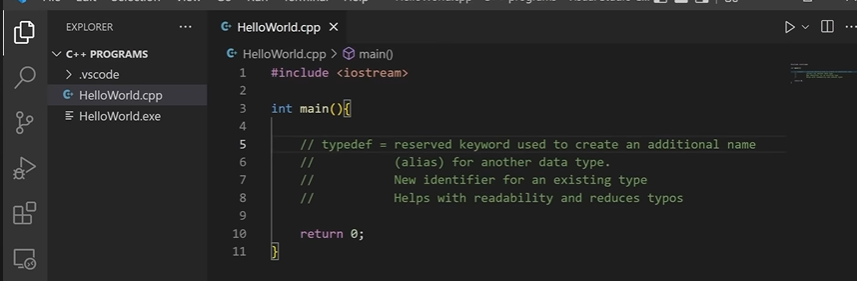
    return 0;

}

Result:



Typedef:



Typedef code:

#include<iostream>

#include<vector>

//typedef std::vector<std::pair<std::string, int>> pairlist\_t;

typedef std::string text\_t;

typedef int number\_t;

int main()

{

    text\_t lastname = "Rangan";

    number\_t age = 43;

    std::cout << lastname << '\n';

    std::cout << age << '\n';

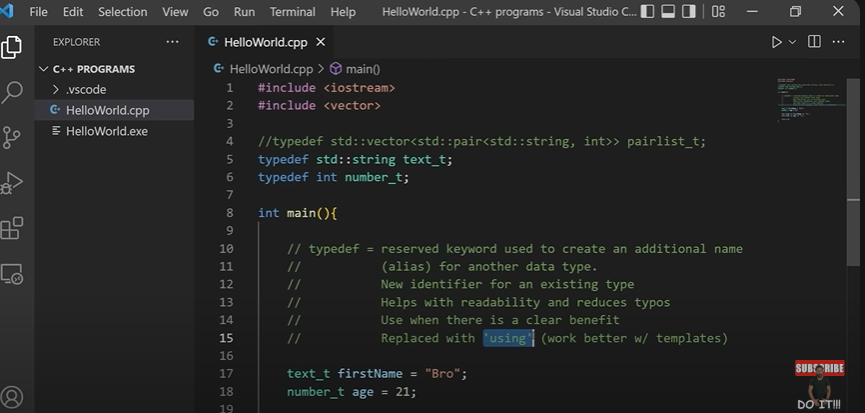
    return 0;

}

Result:



\*\*\*\*\*using the ‘using’ keyword is better than ‘typedef’



Using keyword code:

#include<iostream>

#include<vector>

//typedef std::vector<std::pair<std::string, int>> pairlist\_t;

//typedef std::string text\_t;

//typedef int number\_t;

using text\_t = std::string;

using number\_t = int;

int main()

{

    text\_t lastname = "Rangan";

    number\_t age = 43;

    std::cout << lastname << '\n';

    std::cout << age << '\n';

    return 0;

}

Result:



Arithmetic operators code:

#include<iostream>

int main()

{

    int students = 20;

    //students = students + 2; output = 22

    //students+=2; output = 22

    students++; //output = 21

    std::cout << students << '\n';

    int apples = 5;

    //apples = apples - 2; output = 3

    //apples-=2; output = 3

    apples--; //output = 4

    std::cout << apples << '\n';

    int lemons = 10;

    //lemons = lemons \* 2; output = 20

    lemons\*=2;  //output = 20;

    std::cout << lemons << '\n';

    int babies = 20;

    //babies = babies / 2; output = 10

    //babies /= 2; output = 10

    babies /= 3;  // output = 6

    std::cout << babies << '\n';

    int shirts = 20;

    int remainder = shirts % 3;

    std::cout << remainder << '\n';

    /\*order of precedence of arithmetic operaters

    >parenthesis

    >multiplication & division

    >addition & subtraction   \*/

    int number = 6 - 5 + 4 \* 3 /2;

    std::cout << number << '\n';

    return 0;

}

Result:

