**1.INTRODUCTION TO C++**

1. Write a simple C++ program to display "Hello, World!".

* #include<iostream.h>

Using namespace std;

Int main()

{  
 cout<<”Hello , World”;

Return 0;

}

1. Write a C++ program that accepts user input for their name and age and then displays a personalized greeting.

* .

#include<iostream>

using namespace std;

int main()

{

string name;

int age;

cout <<"enter name and age = ";

cin >>name>>age;

cout<<"hello ,"<<name<<"\t and your age is "<<age<<"nice to meet you";

return 0;

}

1. Write two small programs: one using Procedural Programming (POP) to calculate the area of a rectangle, and another using Object-Oriented Programming (OOP) with a class and object for the same task.

* using procedural programming (pop) to calculate the area of a rectangle:--

#include<stdio.h>

int main()

{

int l,b,area;

printf("\n enter the length and bridth of ractangle : ");

scanf("%d%d",&l,&b);

printf("area of ractangle is %d",l\*b);

return 0;

}

* using oop:--

#include<iostream>

using namespace std;

class area

{

int length, width, ans;

public:

void getdata()

{

cout << "\nenter the length and width: ";

cin >> length >> width;

ans = length \* width;

}

void display()

{

cout << "length = " << length << "\twidth = " << width << "\narea = " << ans << endl;

}

};

int main()

{

area a;

a.getdata();

a.display();

return 0;

}

1. Write a program that asks for two numbers and displays their sum. Ensure this is done after setting up the IDE (like Dev C++ or CodeBlocks).

#include<iostream>

using namespace std;

int main()

{  
 int num1,num2,sum=0;

cout<<”enter num1 and num2 =”;

cin>>num1>>num2;

sum=num1+num2;

cout<<”num1 : “<<num1<<”num2 : “<<num2<<”sum= “<<sum;

return 0;

}

**2. Variables, Data Types, and Operators**

1. Write a C++ program that demonstrates the use of variables and constants. Create variables of different data types and perform operations on them.

-> #include<iostream>

using namespace std;

int main()

{

int n1,n2;

bool temp=true;

char ch;

cout<<"enter number 1 and number 2 : ";

cin>>n1>>n2;

while(temp)

{

cout<<"\n+.addition .";

cout<<"\n-.subtraction .";

cout<<"\n\*.multiplication .";

cout<<"\n/.divistion .";

cout<<"\ne.for exite";

cout<<"\n enter your choice: ";

cin>> ch;

switch(ch)

{

case '+':

cout<<"\naddition of two number is "<<n1+n2;

break;

case '-':

cout<<"\nsubtraction of two number is "<<n1-n2;

break;

case '\*':

cout<<"\nmultiplication of two number is "<<n1\*n2;

break;

case '/':

cout<<"\ndivision of two number is "<<float(n1)/n2;

break;

case 'e':

temp=false;

break;

case 'e':

temp=false;

break;

}

}

return 0;

}

2. Write a C++ program that performs both implicit and explicit type conversions and prints the results.

->

3. Write a C++ program that demonstrates arithmetic, relational, logical, and bitwise operators. Perform operations using each type of operator and display the results.

->

#include<iostream>

using namespace std;

int main()

{

int a, b, ch = 0;

char op;

cout << "enter number 1 and number 2: ";

cin >> a >> b;

while (ch != 5)

{

cout << "\n\n=== main menu ===";

cout << "\n1. arithmetic operators";

cout << "\n2. relational operators";

cout << "\n3. logical operators";

cout << "\n4. bitwise operators";

cout << "\n5. exit";

cout << "\nenter your choice: ";

cin >> ch;

switch (ch)

{

case 1: {

bool arithloop = true;

while (arithloop)

{

cout << "\n\n--- arithmetic operations ---";

cout << "\n+ : addition";

cout << "\n- : subtraction";

cout << "\n\* : multiplication";

cout << "\n/ : division";

cout << "\ne : exit to main menu";

cout << "\nenter your operator: ";

cin >> op;

switch (op)

{

case '+':

cout << "result: " << a + b;

break;

case '-':

cout << "result: " << a - b;

break;

case '\*':

cout << "result: " << a \* b;

break;

case '/':

if (b != 0)

cout << "result: " << float(a) / b;

else

cout << "error: division by zero!";

break;

case 'e':

case 'e':

arithloop = false;

break;

default:

cout << "invalid operator!";

}

}

break;

}

case 2: {

bool relloop = true;

while (relloop)

{

cout << "\n\n--- relational operations ---";

cout << "\na. > (greater than)";

cout << "\nb. < (less than)";

cout << "\nc. >= (greater than or equal)";

cout << "\nd. <= (less than or equal)";

cout << "\ne. == (equal)";

cout << "\nf. != (not equal)";

cout << "\ng. exit to main menu";

cout << "\nenter your choice: ";

cin >> op;

switch (op)

{

case 'a':

cout << a << " > " << b << " = " << (a > b);

break;

case 'b':

cout << a << " < " << b << " = " << (a < b);

break;

case 'c':

cout << a << " >= " << b << " = " << (a >= b);

break;

case 'd':

cout << a << " <= " << b << " = " << (a <= b);

break;

case 'e':

cout << a << " == " << b << " = " << (a == b);

break;

case 'f':

cout << a << " != " << b << " = " << (a != b);

break;

case 'g':

case 'g':

relloop = false;

break;

default:

cout << "invalid choice!";

}

}

break;

}

case 3: {

bool logloop = true;

while (logloop)

{

cout << "\n\n--- logical operations ---";

cout << "\na. && (logical and)";

cout << "\nb. || (logical or)";

cout << "\nc. ! (logical not)";

cout << "\nd. exit to main menu";

cout << "\nenter your choice: ";

cin >> op;

switch (op)

{

case 'a':

cout << "(" << a << " && " << b << ") = " << (a && b);

break;

case 'b':

cout << "(" << a << " || " << b << ") = " << (a || b);

break;

case 'c':

cout << "!a = " << (!a) << ", !b = " << (!b);

break;

case 'd':

case 'd':

logloop = false;

break;

default:

cout << "invalid choice!";

}

}

break;

}

case 4: {

bool bitloop = true;

while (bitloop)

{

cout << "\n\n--- bitwise operations ---";

cout << "\na. & (and)";

cout << "\nb. | (or)";

cout << "\nc. ^ (xor)";

cout << "\nd. ~ (not)";

cout << "\ne. << (left shift)";

cout << "\nf. >> (right shift)";

cout << "\ng. exit to main menu";

cout << "\nenter your choice: ";

cin >> op;

switch (op)

{

case 'a':

cout << a << " & " << b << " = " << (a & b);

break;

case 'b':

cout << a << " | " << b << " = " << (a | b);

break;

case 'c':

cout << a << " ^ " << b << " = " << (a ^ b);

break;

case 'd':

cout << "~" << a << " = " << (~a);

cout << ", ~" << b << " = " << (~b);

break;

case 'e':

cout << a << " << 1 = " << (a << 1);

cout << ", " << b << " << 1 = " << (b << 1);

break;

case 'f':

cout << a << " >> 1 = " << (a >> 1);

cout << ", " << b << " >> 1 = " << (b >> 1);

break;

case 'g':

case 'g':

bitloop = false;

break;

default:

cout << "invalid choice!";

}

}

break;

}

case 5:

cout << "\nexiting program. goodbye!";

break;

default:

cout << "\ninvalid main menu choice!";

}

}

return 0;

}

**3.Control Flow statement**

1. Write a C++ program that takes a student’s marks as input and calculates the grade based on if-else conditions.

->

#include<iostream>

using namespace std;

int main()

{

int rn,total=0;

char name[10],grade;

cout<<"\n enter roll number";

cin>>rn;

cout<<"\n enter name ";

cin>>name;

cout<<"\n enter the marks ";

cin>>total;

if(total>=80)

{

grade='a';

cout<<"\n grade :"<<grade;

}

else if(total<80 && total>=50)

{

grade='b';

cout<<"\n grade :"<<grade;

}

else

{

grade='c';

cout<<"\n grade :"<<grade;

}

return 0;

}

2. Write a C++ program that asks the user to guess a number between 1 and 100. The program should provide hints if the guess is too high or too low. Use loops to allow the user multiple attempts.

->

#include<iostream>

#include<stdlib.h>

#include<ctime>

using namespace std;

int main()

{

cout<<"\n this is the number guessing game ,gusse the number between 1 to 100";

cout<<"\n enter your chance : ";

int ch;

cin>>ch;

srand(time(0));

cout<<"\n now you have only '"<<ch<<"' chance";

int number = (rand()%100)+1,user\_number,i;

for(i=1;i<=ch;i++)

{

cout<<"\n you have "<<i<<" chance";

cout<<"\n enter the number or guess the number : ";

cin>>user\_number;

if(user\_number>number)

{

cout<<"\n to high";

}

else if(user\_number<number)

{

cout<<"\n to low";

}

else

{

cout<<"\n you win.";

break;

}

}

}

3. Write a C++ program to display the multiplication table of a given number using a for loop.

->

#include<iostream>

using namespace std;

int main()

{

int i,table;

cout<<"\n enter the number of table which you want to generate : ";

cin>>table;

for(i=1;i<=10;i++)

{

cout<<table<<" = "<<table<<" \* "<<i<<" = "<<table \* i<<endl;

}

}

1. Write a program that prints a right-angled triangle using stars (\*) with a nested loop.

->

#include <iostream>

using namespace std;

int main() {

int rows;

cout<<"\n enter the row ";

cin >> rows;

for (int i = 1; i <= rows; i++)

{

for (int j = 1; j <= i; j++)

{

cout << "\*";

}

cout << endl;

}

return 0;

}

**4. Functions and Scope**

1. Write a C++ program that defines functions for basic arithmetic operations (add, subtract, multiply, divide). The main function should call these based on user input.

->

#include <iostream>

using namespace std;

int add(int a, int b) {

return a + b;

}

int subtract(int a, int b) {

return a - b;

}

int multiply(int a, int b) {

return a \* b;

}

float divide(float a, float b) {

if (b != 0)

return a / b;

else {

cout << "Error: Division by zero!" << endl;

return 0;

}

}

int main() {

float num1, num2, result;

int choice;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

cout << "\nChoose operation:";

cout << "\n1. Add";

cout << "\n2. Subtract";

cout << "\n3. Multiply";

cout << "\n4. Divide";

cout << "\nEnter your choice (1-4): ";

cin >> choice;

switch(choice) {

case 1:

result = add(num1, num2);

cout << "Result = " << result << endl;

break;

case 2:

result = subtract(num1, num2);

cout << "Result = " << result << endl;

break;

case 3:

result = multiply(num1, num2);

cout << "Result = " << result << endl;

break;

case 4:

result = divide(num1, num2);

cout << "Result = " << result << endl;

break;

default:

cout << "Invalid choice!" << endl;

}

return 0;

}

2. Write a C++ program that calculates the factorial of a number using recursion.

->

#include <iostream>

using namespace std;

int factorial(int n)

{

if (n <= 1)

return 1;

else

return n \* factorial(n - 1);

}

int main() {

int num;

cout << "Enter a number: ";

cin >> num;

if (num < 0)

cout << "Factorial is not defined for negative numbers." << endl;

else

cout << "Factorial of " << num << " is " << factorial(num) << endl;

return 0;

}

3. Write a program that demonstrates the difference between local and global variables in C++. Use functions to show scope.

->

#include <iostream>

using namespace std;

int globalVar = 10;

void showGlobal() {

cout << "Global variable inside function: " << globalVar << endl;

}

void showLocal() {

int globalVar = 5;

cout << "Local variable inside function: " << globalVar << endl;

}

int main() {

cout << "Global variable in main: " << globalVar << endl;

showGlobal();

showLocal();

cout << "Global variable after function call: " << globalVar << endl;

return 0;

}

**5. Arrays and Strings**

1. Write a C++ program that accepts an array of integers, calculates the sum and average, and displays the results.

->

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

int arr[n];

int sum = 0;

float average;

cout << "Enter " << n << " elements:" << endl;

for (int i = 0; i < n; i++) {

cin >> arr[i];

sum += arr[i];

}

average = sum/ float (n);

cout << "Sum = " << sum << endl;

cout << "Average = " << average << endl;

return 0;

}

2. Write a C++ program to perform matrix addition on two 2x2 matrices.

->

#include <iostream>

using namespace std;

int main() {

int matrix1[2][2], matrix2[2][2], result[2][2],i,j;

cout << "Enter elements of first 2x2 matrix:" << endl;

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

{

for(j=0;j<2;j++)

{

cout<<"\n elements ["<<i<<"] "<<"["<<j<<"]";

cin>>matrix1[i][j];

}

}

cout << "Enter elements of second 2x2 matrix:" << endl;

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

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

cin >> matrix2[i][j];

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

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

result[i][j] = matrix1[i][j] + matrix2[i][j];

cout << "Result of matrix addition:" << endl;

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

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

cout << result[i][j] << " ";

cout << endl;

}

return 0;

}

3. Write a C++ program to check if a given string is a palindrome (reads the same forwards and backwards).

->

#include <iostream>

#include <string>

using namespace std;

int main() {

string str, reversed = "";

cout << "Enter a string: ";

cin >> str;

for (int i = str.length() - 1; i >= 0; i--)

reversed += str[i];

if (str == reversed)

cout << "The string is a palindrome." << endl;

else

cout << "The string is not a palindrome." << endl;

return 0;

}

1. **Introduction to Object-Oriented Programming**

1. Write a C++ program that defines a class Calculator with functions for addition, subtraction, multiplication, and division. Create objects to use these functions.

->

#include<iostream>

using namespace std;

class calculation

{

public:

void add(int a, int b)

{

cout<<"addition of two number is "<<a+b;

}

void sub(int a, int b)

{

cout<<"\n subtraction of two number is "<<a-b;

}

void mul(int a,int b)

{

cout<<"\n multiplication of two number is "<<a\*b;

}

void div(int a,int b)

{

cout<<"\n divsion of two number is "<<float(a)/b;

}

};

int main()

{

int num1,num2;

cout<<"\n enter number 1 and number 2 : ";

cin>>num1>>num2;

cout<<"\n1. addition";

cout<<"\n2. subtraction";

cout<<"\n3. multiplication";

cout<<"\n4. division ";

cout<<"\n5. exit";

int ch;

calculation c;

while(ch!=5)

{

cout<<"\n enter your choice";

cin>>ch;

switch(ch)

{

case 1:

c.add(num1,num2);

break;

case 2:

c.sub(num1,num2);

break;

case 3:

c.mul(num1,num2);

break;

case 4:

c.div(num1,num2);

break;

case 5:

break;

}

}

}

2. Create a class BankAccount with data members like balance and member functions like deposit and withdraw. Implement encapsulation by keeping the data members private.

->

#include<iostream>

using namespace std;

class BankAccount

{

int balance=60000;//this var. not directly used by main function

public:

int bl=5000;//this variable can use main function

void display()

{

cout<<"\n Currunt Balance : "<<balance<<endl;//used by this fucntion which is called by onject

}

void deposit(int amount)

{

if(amount<=0)

{

cout<<"\n please enter posit value for deposit ";

}

else

{

balance = balance + amount;

cout<<"\n new Balance : "<<balance;

}

}

void widthraw(int amount)

{

if(amount<=0)

{

cout<<"\n please enter posit value for deposit ";

}

else if(amount<balance)

{

cout<<"\n insufficien balance ";

}

else

{

balance = balance - amount;

cout<<"\n new available balance : "<<balance;

}

}

};

int main()

{

BankAccount b;

b.display();

int amount\_dp;

int ch;

cout<<"\n 1. deposit : ";

cout<<"\n 2. widthraw : ";

cout<<"\n 3. check balance ";

cout<<"\n Enter your choice : ";

cin>>ch;

switch (ch)

{

case 1:

cout<<"\n Enter amount for deposit : ";

cin>>amount\_dp;

b.deposit(amount\_dp);

break;

case 2:

cout<<"\n Enter amount for deposit : ";

cin>>amount\_dp;

b.widthraw(amount\_dp);

break;

case 3:

b.display();

break;

}

}

3. Write a program that implements inheritance using a base class Person and derived classes Student and Teacher. Demonstrate reusability through inheritance.

->

#include <iostream>

using namespace std;

// Base class

class Person {

public:

void showName()

{

cout << "This is a person." << endl;

}

};

// Derived class: Student

class Student : public Person

{

public:

void showStudent() {

cout << "This is a student." << endl;

}

};

// Derived class: Teacher

class Teacher : public Person

{

public:

void showTeacher()

{

cout << "This is a teacher." << endl;

}

};

int main() {

Student s;

Teacher t;

s.showName();

s.showStudent();

t.showName();

t.showTeacher();

return 0;

}