Local and global variables

Local variables:

#include<iostream>

using namespace std;

void show()

{

    cout<<"inside show,c="<<c; //Error

}

int main()

{

int a,b,c;

a=10;

b=20;

c=a+b;

cout<<"Inside main,c="<<c<<endl;

show();

return 0;

}

o/p:

ERROR

Creating local and global variables with the same name:

#include<iostream>

using namespace std;

int c;

void show() //one copy of 'c' global variable

{

    cout<<"inside show,c="<<c;

}

int main()

{

int a,b,c; //2 copies of c:

a=10;

b=20;

c=a+b;

cout<<"Inside main,c="<<c<<endl;

show();

return 0;

}

o/p:

Inside main,c=30

inside show,c=0

Global variables:

#include<iostream>

using namespace std;

int c=100; //global

void show()

{

    cout<<"inside show,c="<<c;

}

int main()

{

cout<<"initially,c="<<c<<endl;

c=1000;

show();

}

o/p:

initially,c=100

inside show,c=1000

using Scope resolution operator(::) to refer the global variables:

#include<iostream>

using namespace std;

int c=1000; //global

void show()

{

    cout<<"inside show,c="<<c;

}

int main()

{

int c=100; //local

cout<<"global var,c="<<::c<<endl;

cout<<"local var,c="<<c<<endl;

show();

}

o/p:

global var,c=1000

local var,c=100

inside show,c=1000

Default values for global variables:

#include<iostream>

using namespace std;

int a;

float b;

char c;

double d;

bool status;

int main()

{

cout<<a<<" "<<b<<" "<<c<<" "<<d<<" "<<status<<endl;

}

o/p:

#include<iostream>

using namespace std;

int a;

float b;

char c;

double d;

int \*p;

bool status;

int main()

{

cout<<a<<" "<<b<<" "<<c<<" "<<d<<" "<<status<<endl;

cout<<"pointer p="<<p<<endl;

}

o/p:

0 0 0 0

pointer p=0

Defining constants in CPP

1. #define
2. Const
3. Enum

Ex:

#include<iostream>

#define PI 3.14

using namespace std;

int main()

{

    int rad=3;

    float area=PI\*rad\*rad;

    cout<<"Area="<<area<<endl;

}

Ex:

#include<iostream>

using namespace std;

int main()

{

    const float PI=3.14;

    int rad=3;

    float area=PI\*rad\*rad;

    cout<<"Area="<<area<<endl;

}

Ex:

#include<iostream>

using namespace std;

int main()

{

    enum color{red,blue,green,yellow};

    color c=green;

    cout<<c;

    return 0;

}

o/p:2

Ex:

#include<iostream>

using namespace std;

int main()

{

    enum color{red=10,blue=20,green=30,yellow=40};

    color c=green;

    cout<<c;

    return 0;

}

o/p:

30

#include<iostream>

using namespace std;

int main()

{

    enum color{red=10,blue,green,yellow};

    color c=green;

    cout<<c;

    return 0;

}

o/p:

12

Ex:

#include<iostream>

using namespace std;

int main()

{

    enum color{red=10,blue,green,yellow} c1,c2,c3;

    c1=red;

    c2=green;

    c3=yellow;

    cout<<c1<<" "<<c2<<" "<<c3;

}

o/p:

10 12 13

CPP strings

#include<iostream>

#include<string>

using namespace std;

int main()

{

string name;

name="CPP Programming";

cout<<name;

}

Indexing to string variables:

#include<iostream>

#include<string>

using namespace std;

int main()

{

string name;

name="CPP Programming";

cout<<name[0];

}

o/p: C

String concatenation:

#include<iostream>

#include<string>

using namespace std;

int main()

{

string fname,lname;

fname="James";

lname="Keringham";

cout<<fname+lname;

}

Append():

#include<iostream>

#include<string>

using namespace std;

int main()

{

string fname,lname;

fname="James";

lname="Keringhan";

fname.append(lname);

cout<<fname<<endl;

}

Strings and numbers:

#include<iostream>

#include<string>

using namespace std;

int main()

{

int x=10;

string name="python";

cout<<x+name;

}

ERROR

Finding the length of a string:

#include<iostream>

#include<string>

using namespace std;

int main()

{

int x=10;

string name="python";

cout<<name.length();

}

Ex: changing a character from a string:

#include<iostream>

#include<string>

using namespace std;

int main()

{

int x=10;

string name="python";

name[0]='J';

cout<<name;

}

o/p:

Jython

Ex: Reading a line of input

#include<iostream>

#include<string>

using namespace std;

int main()

{

int x=10;

string name;

cout<<"enter ur full name:";

getline(cin,name);

cout<<name;

}

Creating references in CPP

#include<iostream>

using namespace std;

int main()

{

int cou=10;

int &check=cou;

cout<<"cou="<<cou<<endl;

cout<<"check="<<check<<endl;

check=1000;

cout<<"cou after updating:"<<cou<<endl;

}

//datatype &var-name=variable;

Default values to parameters in functions in CPP

#include<iostream>

using namespace std;

void sum(int x,int y=100)

{

    int c=x+y;

    cout<<"Sum of "<<x<<" and "<<y<<" is "<<c<<endl;

}

int main()

{

    sum(10,20);

    sum(50);

    //sum(); Error

    return 1;

}

Ex:

#include<iostream>

using namespace std;

void sum(int x=200,int y=100)

{

    int c=x+y;

    cout<<"Sum of "<<x<<" and "<<y<<" is "<<c<<endl;

}

int main()

{

    sum(10,20);

    sum(50);

    sum();

    return 1;

}

Ex:

Assign default values from right to left, not from left to right

#include<iostream>

using namespace std;

void sum(int x=200,int y=100);

void sum(int x,int y=20);

void sum(int x=10,int y);//error

void sum(int x=200,int y)

{

    int c=x+y;

    cout<<"Sum of "<<x<<" and "<<y<<" is "<<c<<endl;

}

int main()

{

    sum(100);

    return 1;

}

Types of function calls

1. Call by value
2. Call by address
3. Call by reference

Call by value:

#include<iostream>

using namespace std;

void sum(int x,int y)

{

    cout<<x+y<<endl;

}

int main()

{

    int a=10,b=20;

    sum(a,b); // call by value

}

Call by address:

#include<iostream>

using namespace std;

void sum(int \*x,int \*y)

{

    cout<<\*x+\*y<<endl;

}

int main()

{

    int a=10,b=20;

    sum(&a,&b); // call by address

}

Ex: swapping numbers using call by value:

#include<iostream>

using namespace std;

void swap(int x,int y)

{

    int temp =x;

    x=y;

    y=temp;

}

int main()

{

    int a,b;

    cout<<"enter a,b values:";

    cin>>a>>b;

    cout<<"Before swapping,a="<<a<<" and b="<<b<<endl;

    swap(a,b); //call by value

    cout<<"After swapping,a="<<a<<" and b="<<b<<endl;

}

Ex: swapping two numbers using call by reference

#include<iostream>

using namespace std;

void swap(int \*x,int \*y)

{

    int temp =\*x;

    \*x=\*y;

    \*y=temp;

}

int main()

{

    int a,b;

    cout<<"enter a,b values:";

    cin>>a>>b;

    cout<<"Before swapping,a="<<a<<" and b="<<b<<endl;

    swap(&a,&b); //call by address

    cout<<"After swapping,a="<<a<<" and b="<<b<<endl;

}

Swapping two numbers using call by reference:

#include<iostream>

using namespace std;

void swap(int &x,int &y)

{

    int temp=x;

    x=y;

    y=temp;

}

int main()

{

    int a,b;

    cout<<"enter a,b values:";

    cin>>a>>b;

    cout<<"Before swapping,a="<<a<<" and b="<<b<<endl;

    swap(a,b); //call by reference

    cout<<"After swapping,a="<<a<<" and b="<<b<<endl;

}

Function Overloading

#include<iostream>

using namespace std;

void sum(int x,int y)

{

    cout<<"Sum="<<x+y<<endl;

}

void sum()

{

    int a,b;

    cout<<"Enter 2 nums:";

    cin>>a>>b;

    cout<<"sum="<<a+b<<endl;

}

void sum(float x,float y)

{

cout<<x+y<<endl;

}

int main()

{

sum(10,20);

sum();

}