***Ngôn ngữ lập trình C/C++***

1. **Môi trường làm việc**

Sử dụng Code Blocks, Dev C++ hoặc Visual Studio C++.

1. ***Tạo một chương trình mới trong Code Blocks***

***File 🡪 New 🡪 Project 🡪 Console Application 🡪 …*** *chọn tên và đường dẫn*

1. ***Chức năng dịch và chạy chương trình***

− Ctrl-F9: Khởi động chức năng dịch và chạy toàn bộ chương trình.

− F9: Chạy chương trình

− F4: Chạy chương trình từ đầu đến dòng lệnh hiện tại (đang chứa con trỏ)

− F7: Chạy từng lệnh một của hàm main(), kể cả các lệnh con trong hàm.

− F8: Chạy từng lệnh một của hàm main(). Khi đó mỗi lời gọi hàm được xem là một lệnh (không chạy từng lệnh trong các hàm được gọi)

− F2 : Bật/tắt cửa sổ lỗi.

1. ***Cấu trúc chương trình C++***

VD :

#include <iostream.h> // phần khai báo nguyên mẫu, thư viện, …

using namespace std;

int main() // tên chương trình chính

{

cout << “hello world”; //lệnh đưa ra màn hình dòng chữ “hello world”

return 0; // thoát khỏi hàm main

}

1. **Biến và kiểu dữ liệu**
2. **Các kiểu dữ liệu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Loại dữ liệu** | **Tên kiểu** | **Số ô nhớ** | **Miền giá trị** |
| Kí tự | Char | 1 byte | -128 … 127 |
|  | Unsigned char | 1 byte | 0…255 |
| Số nguyên | Short | 2 byte | -32768... 32767 |
|  | Int, long | 4 byte | -231…231-1 |
|  | Long long | 8 byte | -263....263-1 |
|  | Unsigned… |  |  |
| Số thực | Float | 4 byte | Biểu diễn 7 chữ số, -1037 … 1037 |
|  | Double | 8 byte | Biểu diễn 14 chữ số, -10307 … 10307 |
| Logic | Bool | 1 byte | True(1) / false(0) |

1. **Khai báo biến.**
   1. **Khai báo không khởi tạo**

tên\_kiểu tên\_biến\_1 ;

tên\_kiểu tên\_biến\_2 ;

…

VD : int x;

Float y;

Double z;

* 1. **Khai báo có khởi tạo**

tên\_kiểu tên\_biến\_1 = gt\_1, tên\_biến\_2 = gt\_2, tên\_biến\_3 = gt\_3 ;

VD : int x = 10; int y =x + 1; double z = 0.5;….

1. **Gán giá trị cho biến**

tên\_biến = biểu thức ;

vd : x = 10; y = 3;

Với ý nghĩa thông thường của phép toán (nghĩa là tính toán và cho lại một giá trị) thì phép toán gán còn một nhiệm vụ nữa là trả lại một giá trị. Giá trị trả lại của phép toán gán chính là giá trị của biểu thức sau dấu bằng. Lợi dụng điều này C++ cho phép chúng ta gán "kép" cho nhiều biến nhận cùng một giá trị bởi cú pháp:

**biến\_1 = biến\_2 = … = biến\_n = gt ;**

với cách gán này tất cả các biến sẽ nhận cùng giá trị gt. Ví dụ:

x=y=z = 10;

1. **Biểu thức**
   1. **Toán tử toán học**

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Tên** | **Ví dụ** |
| + | Cộng | 12 + 4.9 // cho 16.9 |
| - | Trừ | 3.98 - 4 // cho -0.02 |
| \* | Nhân | 2 \* 3.4 // cho 6.8 |
| / | Chia | 9 / 2.0 // cho 4.5 |
| % | Lấy phần dư | 13 % 3 // cho 1 |

* 1. **Toán tử quan hệ**

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Tên** | **Ví dụ** |
| == | So sánh bằng | 5 == 5 // cho 1 |
| != | So sánh không bằng | 5 != 5 // cho 0 |
| < | So sánh hỏ hơn | 5 < 5.5 // cho 1 |
| <= | So sánh hỏ hơn hoặc bằng | 5 <= 5 // cho 1 |
| > | So sánh lớn hơn | 5 > 5.5 // cho 0 |
| >= | So sánh lớn hơn hoặc bằng | 6.3 >= 5 // cho 1 |

* 1. **Toán tử luận lý**

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Tên** | **Ví dụ** |
| ! | Phủ định luận lý | !(5 == 5) // được 0 |
| && | Và luận lý | 5 < 6 && 6 < 6 // được 0 |
| || | Hoặc luận lý | 5 < 6 || 6 < 5 // được 1 |

* 1. **Toán tử trên bit**

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Tên** | **Ví dụ** |
| ~ | Phủ định bit | ~'\011' // được '\366' |
| & | Và bit | '\011' & '\027' // được '\001' |
| | | Hoặc bit | '\011' | '\027' // được '\037' |
| ^ | Hoặc exclusive bit | '\011' ^ '\027' // được '\036' |
| << | Dịch trái bit | '\011' << 2 // được '\044' |
| >> | Dịch phải bit | '\011' >> 2 // được '\002' |

* 1. **Toán tử tăng/giảm**

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Tên** | **Ví dụ (k=5)** |
| ++ | Tăng một (tiền tố) | ++k + 10 // được 16 |
| ++ | Tăng một (hậu tố) | k++ + 10 // được 15 |
| -- | Giảm một (tiền tố) | --k + 10 // được 14 |
| -- | Giảm một (hậu tố) | k-- + 10 // được 15 |

* 1. **Toán tử khởi tạo**

|  |  |  |
| --- | --- | --- |
| **Toán tử** | **Ví dụ** | **Tương đương với** |
| = | n = 25 | |
| += | n += 25 | n = n + 25 |
| -= | n -= 25 | n = n – 25 |
| \*= | n \*= 25 | n = n \* 25 |
| /= | n /= 25 | n = n / 25 |
| %= | n %= 25 | n = n % 25 |
| &= | n &= 15 | n = n & 15 |
| |= | n |= 6 | n = n | 6 |
| ^= | n ^= 13 | n = n ^13 |
| <<= | n <<= 4 | n = n << 4 |
| >>= | n >>= 4 | n = n >> 4 |

* 1. **Toán tử điều kiện**

*toán hạng 1* ? *toán hạng 2* : *toán hạng 3*

VD:

int m = 1, n = 2;

int min = (m < n ? m : n); // min nhận giá trị 1

1. **Lệnh**
   1. **Lệnh nhập xuất**
      1. Nhập

Cú pháp:

cin >> tên biến;

VD : Để nhập dữ liệu cho các biến có tên x,y,z

cin>>x;

cin>>y; …

**note**: *scanf();*

* + 1. Xuất

cout << tên biến;

Vd:

int x = 10, y = 15;

cout<<x; // Màn hình hiện ra số 10

cout<<y; // Màn hình hiện ra số 15

**note**: *printf();*

* 1. **Lệnh điều kiện**
     1. **Lệnh IF**

Đôi khi chúng ta muốn làm cho sự thực thi một lệnh phụ thuộc vào một điều kiện nào đó cần được thỏa. Lệnh if cung cấp cách để thực hiện công việc này, hình thức chung của lệnh này là:

if (*biểu thức*) *lệnh*;

VD: if(a %2 ==0) cout<<”so chan”;

else cout << “so le”

Nhiều lệnh ta dùng {} // khối lệnh

if (*biểu thức*) *lệnh 1*;

else *lệnh 2;*

VD:

if (balance > 0) {

interest = balance \* creditRate;

balance += interest;

} else {

interest = balance \* debitRate;

balance += interest;

}

* + 1. **Lệnh Switch**

switch (*biểu thức*) {

case *hằng1* :  *các lệnh*;

...

case hằng *n: các lệnh*;

default: *các lệnh*;

}

*Biểu thức* (gọi là thẻ switch) được ước lượng trước tiên và kết quả được so sánh với mỗi hằng số (gọi là các nhãn) theo thứ tự chúng xuất hiện cho đến khi một so khớp được tìm thấy. *Lệnh* ngay sau khi so khớp được thực hiện sau đó. Chú ý số nhiều: mỗi case có thể được theo sau bởi không hay nhiều lệnh (không chỉ là một lệnh). Việc thực thi tiếp tục cho tới khi hoặc là bắt gặp một lệnh break hoặc tất cả các lệnh xen vào đến cuối lệnh switch được thực hiện.Trường hợp default ở cuối cùng là một tùy chọn và được thực hiện nếu như tất cả các case trước đó không được so khớp.

switch (operator) {

case '+': result = operand1 + operand2;

break;

case '-': result = operand1 - operand2;

break;

case '\*': result = operand1 \* operand2;

break;

case '/': result = operand1 / operand2;

break;

default: cout << "unknown operator: " << operator << '\n';

break;

}

switch (operator) {

case '+': result = operand1 + operand2;

break;

case '-': result = operand1 - operand2;

break;

case 'x':

case '\*': result = operand1 \* operand2;

break;

case '/': result = operand1 / operand2;

break;

default: cout << "unknown operator: " << operator << '\n';

break;

}

* 1. **Lệnh lặp**

**4.1 Lệnh While**

while (*biểu thức*) *lệnh*;

Đôi khi chúng ta có thể gặp vòng lặp while có thân rỗng (nghĩa là một câu lệnh null). Ví dụ vòng lặp sau đặt n tới thừa số lẻ lớn nhất của nó.

while (n % 2 == 0 && n /= 2) ;

**4.2 Do while**

do

*lệnh*;

while (*biểu thức*);

*Lệnh* được thực thi trước tiên và sau đó *biểu thức* được ước lượng.

Vòng lặp do ít được sử dụng thường xuyên hơn vòng lặp while. Nó hữu dụng trong những trường hợp khi chúng ta cần thân vòng lặp thực hiện ít nhất một lần mà không quan tâm đến điều kiện lặp. Ví dụ, giả sử chúng ta muốn thực hiện lặp đi lặp lại công việc đọc một giá trị và in bình phương của nó, và dừng khi giá trị là 0. Điều này có thể được diễn giải trong vòng lặp sau đây:

do {

cin >> n;

cout << n \* n << '\n';

} while (n != 0);

**4.3 Lệnh For**

for (*biểu thức1*; *biểu thức2*; *biểu thức3*)

*lệnh*;

*Biểu thức1* (thường được gọi là biểu thức khởi tạo) được ước lượng trước tiên. Mỗi vòng lặp *biểu thức2* được ước lượng. Nếu kết quả không là 0 (đúng) thì sau đó *lệnh* được thực thi và *biểu thức3* được ước lượng. Ngược lại, vòng lặp kết thúc. Vòng lặp for tổng quát thì tương đương với vòng lặp while sau:

*biểu thức1*;

while (*biểu thức 2*) {

*lệnh*;

*biểu thức 3*;

}

for (int i = 1; i <= 3; ++i)

for (int j = 1; j <= 3; ++j)

cout << '(' << i << ',' << j << ")\n";

cho tích số của tập hợp {1,2,3} với chính nó, kết quả như sau:

(1,1)

(1,2)

(1,3)…