Sử dụng những kiểu dữ liệu cơ sở trong chương trình

Nhập môn lập trình

Trình bày: …; Email: …@fit.hcmus.edu.vn

Nội dung

- Cấu trúc một chương trình máy tính
- Chương trình đơn giản
- Các kiểu dữ liệu cơ sở và phép toán
- Các hàm thông dụng có sẵn trong thư viện
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

Cấu trúc một chương trình máy tính

Các thành phần chính của chương trình

- Ví dụ

	Chương trình C	Chương trình C++
1	/* Hello.c */	// Hello.cpp
2	#include <stdio.h>	#include <iostream>
3		using namespace std;
4	void main()	void main()
5	{	{
6	pritnf("Hello everybody!");	cout << "Hello everybody!";
7	}	}

  - Khai báo sử dụng các hàm hay đối tượng có sẵn của NNLT (dòng 2)
  - Đầu vào (entry point) của chương trình chính bắt đầu bằng một hàm đặc biệt có tên là **main**, chương trình sẽ bắt đầu chạy tại chỗ này.
  - Chương trình bắt đầu bằng dấu { (dòng 5) và kết thúc bằng dấu } (dòng 7)

Kiểu dữ liệu, hằng, biến

- Ví dụ (chương trình C)

```
#include <stdio.h>

void main()
{
   #define Pi 3.14159          /* hằng số Pi, kiểu dữ liệu float */
   float R = 1.25;             /* biến R, kiểu dữ liệu float */
   float Dientich;             /* biến Dientich, kiểu dữ liệu float */

   Dientich = Pi * R * R;
   printf("Hinh tron, ban kinh = %f\n", R);
   printf("Dien tich = %f", Dientich);
}
```

Kiểu dữ liệu, hằng, biến

- Ví dụ (chương trình C++)

```
#include <iostream>
using namespace std;
void main()
{
   const float Pi 3.14159;    // hằng số Pi, kiểu dữ liệu float
   float R = 1.25;            // biến R, kiểu dữ liệu float
   float Dientich;            // biến Dientich, kiểu dữ liệu float

   Dientich = Pi * R * R;
   cout << "Hinh tron, ban kinh = " << R << endl;
   cout << "Dien tich = " << Dientich;
}
```

Qui ước đặt tên

- Sử dụng kết hợp các chữ cái từ A đến Z, các số từ 0 đến 9, dấu _, bắt đầu bằng chữ cái.
- Tên phải gợi nhớ và có liên quan về mặt ngữ nghĩa với đối tượng được đặt tên.
- Tên có thể được đặt theo qui ước riêng của một số tổ chức, công ty sản xuất phần mềm theo những thỏa thuận cụ thể.

Bộ nhớ và kích thước lưu trữ

- Khi chương trình chạy, mỗi biến hay hằng của chương trình sẽ được kết buộc với một ô nhớ bên trong bộ nhớ của máy tính.
- Tùy theo kiểu dữ liệu, kích thước (hay độ dài) của ô nhớ này (cũng được gọi là kích thước của biến hay hằng tương ứng) sẽ chiếm một số byte nhất định trong bộ nhớ.
- Toán tử sizeof dùng để xác định kích thước của kiểu dữ liệu, biến hay hằng trong C/C++

Bộ nhớ và kích thước lưu trữ

- Ví dụ (chương trình C)

```
#include <stdio.h>

void main()
{
   short Delta=9;
   printf("Kich thuoc bien Delta = %d\n", sizeof(Delta));
   printf("Kich thuoc kieu int = %d\n", sizeof(int));
   printf("Kich thuoc kieu long = %d\n", sizeof(long));
   printf("Kich thuoc kieu float = %d\n", sizeof(float));
   printf("Kich thuoc kieu double = %d\n", sizeof(double));
   printf("Kich thuoc kieu char = %d\n", sizeof(char));
}
```

Bộ nhớ và kích thước lưu trữ

- Ví dụ (chương trình C++)

```
#include <iostream>
using namespace std;
void main()
{
   short Delta=9;
   cout << "Kich thuoc bien Delta = " << sizeof(Delta) << endl;
   cout << "Kich thuoc kieu int = " << sizeof(int) << endl;
   cout << "Kich thuoc kieu long = " <<  sizeof(long) << endl;
   cout << "Kich thuoc kieu float = " <<  sizeof(float) << endl;
   cout << "Kich thuoc kieu double = " <<  sizeof(double) << endl;
   cout << "Kich thuoc kieu char = " << sizeof(char) << endl;
}
```

Chương trình đơn giản

Nhập, xuất, tính toán

- Đa số các chương trình máy tính đều thực hiện ba nhóm thao tác chính như sau:
  - Nhập dữ liệu: nhận dữ liệu từ người sử dụng thông qua thiết bị nhập (bàn phím, chuột, …) hay từ chương trình khác.
  - Tính toán hay xử lý dữ liệu nhập một cách thích hợp để ra được kết quả cần thiết tùy theo bài toán cụ thể.
  - Xuất dữ liệu: gửi kết quả tính toán ra thiết bị xuất (máy in, màn hình, …)

Nhập, xuất, tính toán

- Ví dụ (chương trình C)

```
#include <stdio.h>

void main()
{
   int A, B;                          /* khai báo biến */
   int sum;                           /* khai báo biến */
   printf("Gia tri cua A =");         /* xuất dữ liệu */
   scanf("%d", &A);                   /* nhập dữ liệu */
   printf("Gia tri cua B =");         /* xuất dữ liệu */
   scanf("%d", &B);                   /* khai báo biến */
   sum = A + B;                       /* tính toán, xử lý */
   printf("Tong so = %d\n", sum);     /* xuất dữ liệu */
}
```

Nhập, xuất, tính toán

- Ví dụ (chương trình C)

```
#include <iostream>
using namespace std;
void main()
{
   int A, B;                                  // khai báo biến
   int sum;                                   // khai báo biến
   cout << "Gia tri cua A =";                 // xuất dữ liệu
   cin >> A;                                  // nhập dữ liệu
   cout << "Gia tri cua B =";                 // xuất dữ liệu
   cin >> B;                                  // nhập dữ liệu
   sum = A + B;                               // tính toán, xử lý
   cout << "Tong so = " << sum << endl;       // xuất dữ liệu
}
```

Các kiểu dữ liệu cơ sở và phép toán

Kiểu dữ liệu cơ sở và phép toán

- Các NNLT đều có một hệ thống các kiểu dữ liệu cơ sở cùng với các phép toán để người lập trình có thể thực hiện các tính toán và dựa vào kiểu cơ sở để xây dựng những kiểu dữ liệu phức tạp hơn trong quá trình viết chương trình.
- Các kiểu dữ liệu bao gồm kiểu số nguyên (có dấu và không dấu), kiểu số thực, kiểu luận lý và kiểu ký tự.

Kiểu số nguyên có dấu

- Miền giá trị (số n-bit): -2^{n-1} .. $+2^{n-1} - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
char	1	–128 … +127
int	2 4	–32.768 … +32.767 –2.147.483.648 … +2.147.483.647
short	2	–32.768 … +32.767
long	4	–2.147.483.648 … +2.147.483.647
long long	8	–9,223,372,036,854,775,808 … 9,223,372,036,854,775,807
Một số môi trường lập trình đồng nhất kiểu long long với kiểu long cho nên kiểu này ít được sử dụng trong lập trình ứng dụng.		

Kiểu số nguyên không dấu

- Miền giá trị (số n-bit): 0 .. $2^n - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
unsigned char	1	0 … 255
unsigned int	2 4	0 … 65535 0 … 4.294.967.295
unsigned short	2	0 … 65535
unsigned long	4	0 … 4.294.967.295
unsigned long long	8	0 … 18,446,744,073,709,551,615
Một số môi trường lập trình đồng nhất kiểu unsigned long long với kiểu unsigned long cho nên kiểu này ít được sử dụng trong lập trình ứng dụng.		

Kiểu số nguyên

- Hằng số nguyên có thể biểu diễn ở 3 dạng
  - Bát phân: viết bắt đầu bằng số 0
  - Thập phân: viết bắt đầu bằng số từ 1 đến 9
  - Thập lục phân: viết đầu bằng 0x
- Các phép toán số học
  - Phép cộng: +, phép trừ: –, phép nhân: *
  - Phép chia lấy phần nguyên: /
  - Phép chia lấy phần dư: %

Kiểu số nguyên

- Các phép toán trên bit cho số nguyên không dấu (được áp dụng khi muốn lập trình thao tác trên các bit của dữ liệu hay muốn tăng tốc độ xử lý của chương trình trong một vài tình huống nhất định)
  - Phép and bit: &
  - Phép or bit: |
  - Phép xor bit: ^
  - Phép not bit: ~

Ví dụ toán tử trên bit

```
#include <stdio.h>

void main()
{
   unsigned char a = 45;    // 00101101
   unsigned char b = 58;    // 00111010

   int c1, c2, c3, c4, c5, c6;
   c1 = a & b;              // 00101000
   c2 = a | b;              // 00111111
   c3 = a ^ b;              // 00010111
   c4 = ~a;                 // 11010010
   c5 = a >> 4;             // 11010000
   c6 = a << 4;             // 00000010
}
```

Kiểu số thực

- Cấu trúc lưu trữ bên trong của số thực được thiết kế theo chuẩn số chấm động (floating-point) của IEEE.

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
float	4	$1,4x10^{-45}$ ... $3,4x10^{38}$
float có độ chính xác đơn (single-precision), chính xác đến 7 chữ số.		
double	8	$4,94x10^{-324}$ ... $1,79x10^{308}$
double có độ chính xác kép (double-precision), chính xác đến 15 chữ số.		
long double	10	... $3,4x10^{4932}$
Một số môi trường lập trình đồng nhất kiểu long double với kiểu double cho nên kiểu này ít được sử dụng trong lập trình ứng dụng.		

Kiểu số thực

- Các phép toán số học
  - Phép cộng: +
  - Phép trừ: –
  - Phép nhân: *
  - Phép chia: /
- Các hàm toán học như căn số, lũy thừa, logarit, … sẽ được trình bày ở phần sau.

Kiểu luận lý

- Khai báo kiểu bool đối với C++ chuẩn hoặc kiểu số nguyên bất kỳ (char, int, …)
  - Giá trị khác 0 nghĩa là đúng (true).
  - Giá trị bằng 0 nghĩa là sai (false).
  - *Lưu ý: Kết quả lượng giá một biểu thức luận lý bất kỳ thực hiện bởi C++ luôn cho kết quả là 0 (false) hay 1 (true).*
- Các phép toán
  - Kết hợp: && (and), || (or), ! (not)
  - So sánh: >, >=, <, <=, ==, !=

Ví dụ

```
#include <stdio.h>

void main()
{
   bool bVal;
   double x=46.7, y=93, z;
   bVal = (x==y);
   printf("%d\n", bVal);
   bVal = (x<y);
   printf("%d\n", bVal);
   bVal = (2*x>y);
   printf("%d\n", bVal);
   z = (x>y)*x + (x<=y)*y;
   printf("%f\n", z);
}
```

```
#include <iostream>
using namespace std;
void main()
{
   bool bVal;
   double x=46.7, y=93, z;
   bVal = (x==y);
   cout << bVal << endl;
   bVal = (x<y);
   cout << bVal << endl;
   bVal = (2*x>y);
   cout << bVal << endl;
   z = (x>y)*x + (x<=y)*y;
   cout << z << endl;
}
```

Kiểu ký tự

- Kiểu ký tự 8-bit
  - Kiểu char hoặc unsigned char.
  - Lưu mã ASCII của ký tự, giá trị từ 0 đến 255.
  - Một số ký tự nên nhớ

Ký tự	Mã
' ' (khoảng trắng)	32
'0' .. '9'	48 .. 57
'A' .. 'Z'	65 .. 90
'a' .. 'z'	97 .. 122

Kiểu ký tự

- Đổi ký tự từ ký tự thường sang ký tự hoa:
  - Nếu ‘a’ ≤ ch ≤ ’z’ thì ch (mới) = ch – (‘a’ – ‘A’)
  - Ngược lại ch (mới) = ch
- Trong mọi trường hợp ta có công thức:
  - ch – (‘a’ – ‘A’) * (ch >= ‘a’ && ch <= ‘z’)
- Tương tự ta cũng có công thức chuyển ký tự thành ký tự thường:
  - ch – (‘a’ – ‘A’) * (ch >= ‘A’ && ch <= ‘Z’)

Ví dụ

```
#include <stdio.h>

void main()
{
   char ch;
   ch=65;
   printf("ch = %c\n", ch);
   ch = 'A';
   printf("ch = %c\n", ch);
   printf("ch = ");
   scanf("%c", &ch);
   printf("ASCII code = %d\n", ch);
   ch -= ('a' – 'A')*(ch>='a' && ch<=z);
   printf("Upper case: %c\n", ch);
}
```

```
#include <iostream>
using namespace std;
void main()
{
   char ch;
   ch=65;
   cout << "ch = " << ch << endl;
   ch = 'A';
   cout << "ch = " << ch << endl;
   cout << "ch = ";
   cin >> ch;
   cout <<"ASCII code = " << ch << endl;
   ch -= ('a' – 'A')*(ch>='a' && ch<=z);
   cout << "Upper case: " << ch << endl;
}
```

Kiểu ký tự

- Kiểu ký tự 16-bit
  - Kiểu wchar_t (#include <wchar>)
  - Lưu trữ dựa trên bảng mã quốc tế UTF-16 (một dạng mã Unicode)
    - Mã UTF-16 của ký tự thông thường ('0' đến '9', 'A' đến 'Z', 'a' đến 'z', ...) trùng mã ASCII.
  - Hằng ký tự kiểu wchar_t được đặt trước bằng chữ L
    - Lưu ý, 'B' và L'B' như nhau (cùng giá trị 66) nhưng kích thước trong bộ nhớ khác nhau (sizeof('B') = 1, sizeof(L'B') = 2)

Phép gán

- Việc tính toán trong chương trình được thực hiện bằng cách tính toán và chép kết quả tính toán vào một biến nằm bên trái của phép gán.
- Ví dụ:

```
sum = A + B;    // chép tổng A + B vào biến sum
sum = A + 2;    // chép tổng A + 2 vào biến sum
sum = A + n;    // chép tổng A + n vào biến sum
```

Lệnh viết ngắn

- Ví dụ:
  - Viết sum++ (hay ++sum) thay cho sum = sum + 1;
  - Viết sum += 2 thay cho sum = sum + 2;
  - Viết sum += n thay cho sum = sum + n;
  - Viết n <<= 2 thay cho n = n << 2;
  - Viết n = m++ tương đương với n = m; rồi m++;
  - Viết n = ++m tương đương với ++m rồi n = m;
- Việc viết các lệnh cô đọng có thể làm cho chương trình khó đọc, khó bắt lỗi vì vậy không nên lạm dụng!

Định dạng dữ liệu nhập xuất

- Đối với NNLT C
  - Nhập xuất số nguyên (kiểu char, int)
    - Có dấu dạng thập phân: %d hay %i
    - Không dấu:
      - Dạng thập phân: %u
      - Dạng thập lục phân: %x hay %X
      - Dạng bát phân: %o
  - Trường hợp nhập xuất số nguyên kiểu khác:
    - short (16-bit): %hd, %hi, %ho, %hu, %hx, %hX
    - long: %ld, %li, %lo, %lu, %lx, %lX

Định dạng dữ liệu nhập xuất

- Đối với NNLT C
  - Nhập xuất số thực chấm động (kiểu float)
    - Dạng viết thập phân: %f
    - Dạng viết số mũ (chữ e hay E thay cho cơ số 10, ví dụ 1.2E-8): %e hay %E
  - Trường hợp nhập xuất thực kiểu khác:
    - double: %lf, %le, %lE
    - long double: %Lf, %Le, %LE

Định dạng dữ liệu nhập xuất

- Đối với NNLT C
  - Ký tự đặc biệt: \\ (dấu \) và %% (dấu %)
  - Ký tự tab và ký tự xuống dòng: \t, \n, \r
  - Nhập xuất ký tự: %c
  - Nhập xuất chuỗi ký tự: %s
  - Về việc quy định động rộng và độ chính xác (nếu là số thực) cho dữ liệu xuất được ghi ngay sau dấu % với dạng wid.pre, ví dụ %9.2f nghĩa là độ rộng ít nhất 9 ký tự (thêm khoảng trống vào nếu thiếu) và nhiều nhất là 2 ký tự cho phần lẻ sau dấu chấm thập phân.

Định dạng dữ liệu nhập xuất

- Đối với NNLT C++
  - Việc nhập xuất được thực hiện bởi các đối tượng đã được định nghĩa sẵn trong <iostream>:
    - cin kèm với toán tử >> (được gọi là extraction operator) để nhập dữ liệu.
    - cout kèm với toán tử << (được gọi là insertion operator) để xuất dữ liệu.
  - Được cung cấp hệ thống định dạng dữ liệu nhập xuất cho thiết bị nhập chuẩn và mở rộng cho các thiết bị nhập xuất khác như tập tin.

Định dạng dữ liệu nhập xuất

- Đối với NNLT C++
  - Thêm chỉ thị sau vào đầu chương trình:

```
#include <iomanip>
```

  - Việc định dạng dữ liệu được thực hiện bằng các toán tử định dạng (manipulator).
    - **endl**: xuống dòng mới.
    - **setw(n)**: định độ rộng của dữ liệu xuất.
    - **left** và **right**: dùng chung với setw(n) để canh lề trái hay lề phải.
    - **setfill(ch)**: dùng chung với setw(n) để qui định ký tự ch được thêm vào thay vì dùng khoảng trắng.
    - **dec**, **oct**, **hex**: được dùng để qui định số nguyên (khi nhập xuất) được ghi theo dạng thập phân, bát phân, thập lục phân.
    - **setprecision(n)**: dùng để qui định độ chính xác khi in số thực.

Ví dụ

```
#include <iostream>
#include <iomanip>
using namespace std;
void main()
{
   int Area=970, Height=10, Volume=9700;
   cout << setw(8) << "Area" << setw(10) << Area << endl;
   cout << setw(8) << "H" << setw(10) << Height << endl;
   cout << setw(8) << "Volume" << setw(10) << Volume << endl;
}
```

Kết quả chạy chương trình

```
    Area       970
       H        10
  Volume      9700
```

Ví dụ

```
#include <iostream>
#include <iomanip>
using namespace std;
void main()
{
   long n;
   cout << "n (hexadecimal) = ";
   cin >> hex >> n;
   cout << "Octal representation: " << oct << n << endl;
   cin >> n;
}
```

Độ lớn, độ chính xác, vấn đề tràn số (overflow)

- Đọc thêm trong giáo trình Nhập môn lập trình, Chương 2 – Phần III.6, trang 49-56.

Các hàm thông dụng có sẵn trong thư viện

Hàm và thư viện hàm

- Khái niệm
  - Để tiết kiệm công sức, người lập trình có thể sử dụng lại các hàm (đoạn chương trình) có sẵn trong quá trình viết chương trình ví dụ như tính căn số, lũy thừa, trị tuyệt đối, logarit, …
  - Tập hợp các hàm được xây dựng sẵn của NNLT thường được gọi là thư viện hàm.
  - Hệ thống thư viện hàm rất đa dạng nên người lập trình cần phải tra cứu thêm tài liệu tham khảo hoặc hệ thống giúp đỡ của phần mềm hỗ trợ lập trình.

Ví dụ tính $F(x, y) = x + \sqrt{1 + y^2}$

```c
#include <math.h>
#include <stdio.h>

void main()
{
   double x, y, Fxy;
   printf("x = ");
   scanf("%lf", &x);
   printf("y = ");
   scanf("%lf", &y);
   Fxy = x + sqrt(1 + y*y);
   printf("F(x, y) = %lf", Fxy);
}
```

```c
#include <cmath>
#include <iostream>
using namespace std;
void main()
{
   double x, y, Fxy;
   cout << "x = ";
   cin >> x;
   cout << "y = ";
   cin >> y;
   Fxy = x + sqrt(1 + y*y);
   cout << "F(x, y) = ", Fxy);
}
```

Các hàm toán học

- Các hàm toán học đa số có tham số kiểu double, giá trị nhập vào và kết quả tính toán đều có kiểu double.
- Để sử dụng các hàm toán học, người lập trình cần ghi thêm vào đầu chương trình chỉ thị:
  - #include <math.h> đối với NNLT C
  - #include <cmath> đối với NNLT C++ chuẩn

Các hàm toán học

- Một số hàm toán học thông thường

Nguyên mẫu hàm	Công dụng
double sqrt(double x);	Tính $\sqrt{x}$
double pow(double x, double y);	Tính x^y $(x > 0)$
double exp(double x);	Tính e^x $(e \approx 2{,}71828)$
double log(double x);	Tính $\ln(x)$
double log10(double x);	Tính $\log_{10}(x)$

int abs(int x);
long labs(long x);
double fabs(double x);
	Tính $\|x\|$ (x kiểu int)
	Tính $\|x\|$ (x kiểu long)
	Tính $\|x\|$ (x kiểu double)

Các hàm toán học

- Một số hàm toán học thông thường

Nguyên mẫu hàm	Công dụng

double cos(double x);
double sin(double x);
double tan(double x);
	Tính $\cos(x)$, $\sin(x)$, $\tan(x)$ (x tính theo radian, 1 radian bằng $180/\pi$ độ)

double acos(double x);
double asin(double x);
double atan(double x);
	Tính $\cos^{-1}(x)$
	Tính $\sin^{-1}(x)$
	Tính $\tan^{-1}(x)$

double floor(double x); double ceil(double x);	Tính $[x]$ Tính $[x]$

Các hàm ký tự

- Để sử dụng các hàm ký tự liệt kê trong danh sách sau bằng cách dùng thư viện nhờ chỉ thị #include <ctype.h>

Nguyên mẫu hàm	Công dụng
bool isupper(char ch); bool iswupper(wchar_t ch);	Kiểm tra ch có phải là ký tự hoa?
char toupper(char ch); wchar_t towupper(wchar_t ch);	Trả về ký tự hoa tương ứng với ch
bool islower(char ch); bool iswlower(wchar_t ch);	Kiểm tra ch có phải là ký tự thường?
char tolower(char ch); wchar_t towlower(wchar_t ch);	Trả về ký tự thường tương ứng với ch

Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp

Đọc thêm

- Lịch sử phát triển dữ liệu cơ sở theo NNLT
- Chuẩn lưu trữ vật lý của các loại dữ liệu cơ sở
- Lỗ hổng bảo mật trong mã nguồn
- Sự khác biệt, tương đồng giữa các NNLT

Thuật ngữ
và bài đọc thêm tiếng Anh

Thuật ngữ tiếng Anh

- ***ASCII code***: mã ký tự theo chuẩn 1 byte. Bảng mã ASCII (American Standard Code for Information Interchange) có 256 ký tự (gồm cả ký tự thông thường và ký tự đặc biệt)
- ***character***: ký tự nói chung
  - ***wide character***: ký tự 16 bit
  - ***wide string***: chuỗi ký tự gồm các ký tự 16 bit
- ***constant***: hằng số
- ***data type***: kiểu dữ liệu
- ***floating-point, real data type***: số thực dấu chấm động, kiểu dữ liệu số thực
- ***functtion library***: thư viện hàm
- ***fundamental data type***: kiểu dữ liệu cơ bản, cơ sở
- ***input***: nhập
  - ***input data***: dữ liệu nhập

Thuật ngữ tiếng Anh

- ***integral data type, integer***: kiểu dữ liệu nguyên
  - ***long integer***: kiểu nguyên dài (32 bit)
- ***operator***: toán tử, phép toán
  - ***bit mask***: mặt nạ bit
  - ***bit operator***: phép toán trên bit
  - ***logical operator, boolean operator***: phép toán luận lý
- ***ouput***: xuất
  - ***ouput data***: dữ liệu xuất
- ***overflow***: tràn số
- ***unicode***: nói chung về ký tự unicode
- ***variable***: biến (dùng trong lập trình)
  - ***variable declaration***: khai báo biến

Bài đọc thêm tiếng Anh

- **Thinking in C**, Bruce Eckel, E-book, 2006.
- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.