BÀI TẬP VỀ NHÀ 2

1. Cho biết kết quả của đoạn code vừa biên dịch (2điểm)?
2. Nhận xét kết quả của 2 đoạn code sau và giải thích vì sao dòng 2,3 của mỗi đoạn in ra kết quả như vậy (3điểm)?

Câu 1:

#include <stdio.h>  
#include <math.h>  
#include <limits.h>  
  
void main()  
{  
 printf("%u\n",UINT\_MAX);  
 printf("%u\n", UINT\_MAX + 1);  
 printf("%u\n",UINT\_MAX+2);  
  
 printf("%d\n",INT\_MAX);  
 printf("%d\n",INT\_MAX+1);  
 printf("%d\n",INT\_MAX+2);  
  
 float x = 0.1 + 0.1 + 0.1;  
 printf("%.50f",x);

Kết quả đoạn code

Text

Description automatically generated

Câu 2:

1. Giải thích 3 dòng code đầu

UINT\_MAX; INT\_MAX là các biến constant 16 bit

Với UINT\_MAX: tất cả các bit được bật lên 1 vì là unsign nên cả 16 bit được dùng để thể hiện số

Các biến này gáng giá trị vào bộ nhớ tương ứng với giá trị được giải thích phía trên là:

UINT\_MAX = [1111 1111 1111 1111 1111 1111 1111 1111] = 4294967295

Vì trình biên dịch với định dạng “**%u**” sẽ biên dịch là số nguyên không dấu với 16bits bộ nhớ khi cộng 1 và cộng 2 sẽ sảy ra như sau:

UINT\_MAX = [1111 1111 1111 1111 1111 1111 1111 1111]

+

[0000 0000 0000 0000 0000 0000 0000 0001]

Kết quả : [1] [0000 0000 0000 0000 0000 0000 0000 0000]

1 bít bị out ra khỏi vùng nhớ 16 bit khi biên dịch với format “**%u**” sẽ ra kết quả là “0”.

Tương tự như vậy khi cộng với 2 ([0000 0000 0000 0010]) kết quả sẽ ra [0000 0000 0000 0000 0000 0000 0000 0001] => kết quả sẽ ra giá trị là “0”.

UINT\_MAX = [1111 1111 1111 1111 1111 1111 1111 1111]

+

[0000 0000 0000 0000 0000 0000 0000 0010]

Kết quả : [1] [0000 0000 0000 0000 0000 0000 0000 0001]

khi biên dịch 16bits bộ nhớ với format “**%u**” sẽ ra kết quả là “1”.

1. Giải thích 3 dòng code sau:

Với INT\_MAX: 15 bit được bật lên 1 trừ bit bên trái ngoài cùng (vì là bit xác định dấu của giá trị và là const max nên chỉ 15 bit được bật lên 1)

Với biến const INT\_MAX: bộ nhớ được lưu 16bit với dạng như sau:

INT\_MAX = [0111 1111 1111 1111 1111 1111 1111 1111] = 2147483647

Khi INT\_MAX + 1:

INT\_MAX = [0111 1111 1111 1111 1111 1111 1111 1111]

+

[0000 0000 0000 0000 0000 0000 0000 0001]

Kết quả : [1000 0000 0000 0000 0000 0000 0000 0000]

Bù 1 sẽ là [**1**111 1111 1111 1111 1111 1111 1111 1111]

Bù 2 sẽ là [**1**000 0000 0000 0000 0000 0000 0000 0000] khi biên dịch 16bits bộ nhớ với format “**%d**” sẽ ra kết quả là = -2147483648

Tương tự với INT\_MAX +2:

INT\_MAX = [0111 1111 1111 1111 1111 1111 1111 1111]

+

[0000 0000 0000 0000 0000 0000 0000 0010]

Kết quả : [1000 0000 0000 0000 0000 0000 0000 0001]

Bù 1 sẽ là [**1**111 1111 1111 1111 1111 1111 1111 1110]

Bù 2 sẽ là [**1**111 1111 1111 1111 1111 1111 1111 1111]

khi biên dịch 16bits bộ nhớ với format “**%d**” sẽ ra kết quả là = -2147483647.

1. Vì sao 0.1+0.1+0.1 không bằng 0.3 (2điểm)?

Giá trị float 0.1 là dạng không thể chuẩn hoá được, được biển diễn gần đúng và không thể biển diễn chính xác giá trị 0.1:

Với giá trị 0.1 được chuyển thành binary

0.1d = 00111101110011001100110011001101b

với sai số = **1.490116119384765625E-9**. Và giá trị thực sự được lưu trong bộ nhớ là:

0.10000000**1490116119384765625**

Khi thực hiện phép cộng, bằng bit sau đó biên dịch ngược kết qủa từ bit

điều này khiến cho việc cộng 3 số 0.1 + 0.1 + 0.1 khiến kết quả sai lệt không ra kết quả chính xác được.

1. Vì sao 1.3\*0.25 không bằng 0.325 (2điểm)?

Tương tự như câu 3 với số 1.3 không thể chuẩn hóa được và binary không thể thể hiện chính xác số mà dữ liệu được lưu binary với kết quả: 1.2999999523162841796875 với sai số là -4.76837158203125E-8

Với 1.3d = 00111111101001100110011001100110b

Khiến kết quả tính toán được không ra chính xác.

1. Hiệu chỉnh đoạn code trên để in ra giá trị của phép tính **float z = 1/3**

Code:

#include <stdio.h>  
#include <math.h>  
#include <limits.h>  
  
void main()  
{  
 printf("%u\n",UINT\_MAX);  
 printf("%u\n", UINT\_MAX + 1);  
 printf("%u\n",UINT\_MAX+2);  
  
 printf("%d\n",INT\_MAX);  
 printf("%d\n",INT\_MAX+1);  
 printf("%d\n",INT\_MAX+2);  
  
 float x = 0.1 + 0.1 + 0.1;  
 printf("%.50f",x);  
  
 float y = 1.3 \* 0.25;  
 printf("\n%.50f",y);  
 float z = 1.0 / 3.0 ;  
 printf("\nkết quả z = 1/3 = %.50f",z);  
 return;  
}

Kết quả hiệu chỉnh code:Text

Description automatically generated