Zadania

Zad. 1.1

Napisz program `size` sprawdzający, ile bajtów zajmują typy: `char`, `short`, `int`, `long`, `long int`, `long long` oraz `float`, `double`, `long double`.

`https://en.wikipedia.org/wiki/C_data_types`

Zad. 1.2 *

Napisz program `size2` sprawdzający, ile bajtów zajmują typy: `char`, `short`, `int`, `long`, `long int` i `long long` bez znaku.

Zad. 1.3 *

Napisz program `limits` wypisujący maksymalne wartości, jakie mogą przechowywać typy bez znaku: `char`, `short`, `int`, `long`, `long long` oraz minimalne i maksymalne wartości dla tych samych typów ze znakiem oraz dla typów `float`, `double`, `long double`. Przykład:

```
UCHAR_MAX = 255

 CHAR_MIN = -128
 CHAR_MAX = 127
```

Zad. 1.4

Która z poniższych odpowiedzi jest prawdziwa:

- system 32 bitowy pozwala na uruchamianie programu 32 bitowego
- system 32 bitowy pozwala na uruchamianie programu 64 bitowego
- system 64 bitowy pozwala na uruchamianie programu 32 bitowego
- system 64 bitowy pozwala na uruchamianie programu 64 bitowego

Zad. 1.5 *

Która z poniższych odpowiedzi jest fałszywa:

- na systemie 32 bitowym można skompilować program do kodu 32 bitowego
- na systemie 32 bitowym można skompilować program do kodu 64 bitowego
- na systemie 64 bitowym można skompilować program do kodu 32 bitowego
- na systemie 64 bitowym można skompilować program do kodu 64 bitowego

Zad. 1.6

Napisz program `bits` rozpoznający do ilu bitowego kodu został skompilowany.

Zad. 1.7

Napisz program `process` umieszczający na stosie kolejno dwie zmienne `x` i `y` typu `int` oraz odczytaj adresy tych zmiennych. Skopiuj poniższy schemat do komentarza w programie oraz wypełnij go odpowiednimi wartościami. Czy adresy zmiennych są zgodne z mapą pamięci dla procesu?

```
var1 [ ][ ][ ][ ]   &var1 = addr1
var2 [ ][ ][ ][ ]   &var2 = addr2
```

Zad. 1.8 *

W programie `process` umieść w sekcji danych kolejno dwie zmienne `a` i `b` typu `int` oraz odczytaj adresy tych zmiennych. Rozpatrz następujące przypadki:

- zmienne `a` i `b` są zainicjowane
- zmienne `a` i `b` są niezainicjowane
- zmienna `a` jest zainicjowana i zmienna `b` jest niezainicjowana
- zmienna `a` jest niezainicjowana i zmienna `b` jest zainicjowana

Czy adresy zmiennych są zgodne z mapą pamięci dla procesu?

Zad. 1.9 *

W programie `process` umieść na stercie dwie zmienne `c` i `d` typu `int` oraz odczytaj adresy tych zmiennych. Czy adresy zmiennych są zgodne z mapą pamięci dla procesu? Na ile sposobów można rozwiązać to zadanie?

Zad. 1.10

Załóżmy, że typ `int` zajmuje `4` bajty. Na ile sposobów można umieścić w pamięci pod adresem `p` wartość `1` typu `int`?

```
p -> [ ][ ][ ][ ]   *p = 1
```

Zad. 1.11

Procesory w architekturze `little-endian` czytają młodsze bajty (LSB – least significant byte) od prawej do lewej. Procesory w architekturze `big-endian` czytają starsze bajty (MSB – most significant byte) od lewej do prawej. Załóżmy, że pod adresem `p` znajduje się liczba 5 typu `int`. Wypełnij komórki pamięci odpowiednimi wartościami dla obu architektur.

```
little-endian
```

```
p -> [ ][ ][ ][ ]   *p = 5

big-endian

p -> [ ][ ][ ][ ]   *p = 5
```

Zad. 1.12

W pliku `szereg.txt` rozwiń w szereg i wyznacz wartości dziesiętne dla liczb:

```
1011  – liczba binarna
8732  – liczba dziesiętna
[2][2][1][1]  - reprezentacja bajtowa liczby typu int *

1234  - liczba ósemkowa *
3A5B  – liczba szesnastkowa *
ZZTOP - liczba w systemie opartym o alfabet angielski (A..Z) *
*) w alfabetach z literami wielkość liter nie ma znaczenia
```