Lista zadań nr 2

Rekurencja, dynamiczna alokacja pamięci dla tablic dwuwymiarowych.

Zadania podstawowe:

Zadanie 1 Utwór funkcję rekurencyjną `power(base, exponent)`, która po wywołaniu zwróci: $\texttt{base}^{\texttt{exponent}}$. Na przykład `power(3, 4) = 2 * 3 * 3 * 3`. Przyjmij założenie, że `exponent` to liczba całkowita większa lub równa 1. Wskazówka: krok rekurencyjny powinien używać następującego wyrażania: $\texttt{base}^{\texttt{exponent}} = \texttt{base} * \texttt{base}^{\texttt{exponent - 1}}$, a warunek kończący rekurencję zachodzi gdy, wartość `exponent` wynosi 1, ponieważ `base = base`.

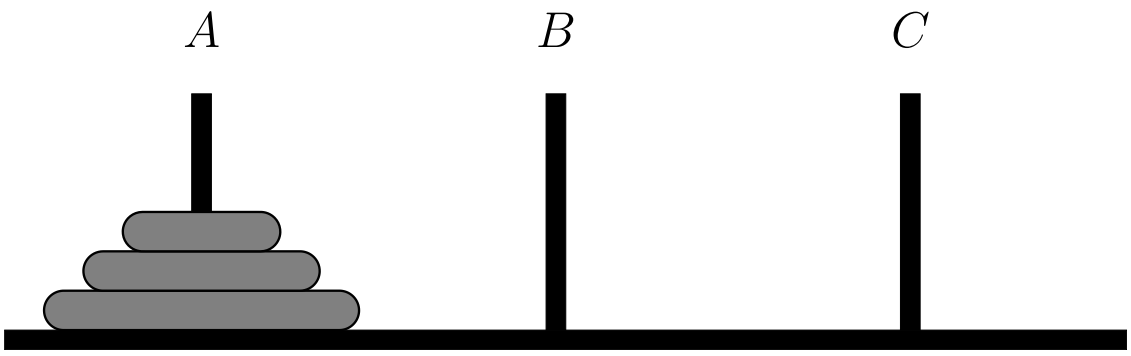
Zadanie 2 Największy wspólny dzielnik liczb całkowitych x oraz y $(x \leqslant y)$ to największa liczba całkowita, która bez reszty dzieli obie te liczby. Utwórz funkcję rekurencyjną `gcd()`, która zwraca największy wspólny dzielnik dwóch liczb całkowitych. Ta funkcja powinna zostać zdefiniowana następująco: jeżeli wartość x wynosi 0, wówczas wynikiem wywołania `gcd(x, y)` jest y. W przeciwnym wypadku wynikiem jest `gcd(y % x, x)`, gdzie % to operator reszty z dzielenia.

Zadanie 3 Jednym z klasycznych przykładów problemów rekurencyjnych jest przykład tzw. *wież Hanoi* (zob. rysunek 1). Legenda głosi, że w świątyni znajdującej się gdzieś na Dalekim Wschodzie grupa mnichów próbuje przenieść krążki z jednego słupka na drogi. Początkowo słupek zawiera 64 krążki ułożone jeden na drugim, od największego na dole do najmniejszego na górze. Podczas przenoszenia krążków mnisi muszą przestrzegać pewnych zasad: dozwolone jest przenoszenie tylko jednego krążka w danej chwili i nigdy większy krążek nie może zostać umieszczony na mniejszym. Do tymczasowego przechowywania krążków używany jest trzeci słupek. Legenda głosi, że po przeniesieniu krążków z jednego słupka na drugi nastąpi koniec świata. Można więc powiedzie, że nie ma zbyt wielkiej motywacji, aby wspomóc wysiłków mnichów. Przyjmijmy założenie, że mnisi próbują przenieść krążki ze słupka pierwszego (A) na trzeci (C). Zadanie polega na opracowaniu algorytmu pozwalającego na wyświetlenie sekwencji przenoszenia krążków.

Jeżeli do rozwiązywania tego problemu zostanie użyta metoda konwencjonalna, bardzo szybko można zabrnąć w ślepy zaułek. Natomiast jeżeli zastosujemy rekurencję, zadanie stanie się możliwe do wykonywania. Przesuniecie n krążków można postrzegać w kategoriach przesunięcia jedynie $n-1$ krążków (stąd rekurencja), zgodnie z następującym opisem:

a) przeniesienie $n-1$ krążków ze słupka A na B z użyciem słupka C jako przeznaczonego do tymczasowego przechowywania krążków;

b) przeniesienie ostatniego krążka (największego) ze słupka A na C;



Rys. 1: Wieże Hanoi i trzy krążki.

c) przeniesienie $n-1$ krążków ze słupka B na C z użyciem słupka A jako przeznaczonego do tymczasowego przechowywania krążków.

Ten proces zakończy się, gdy ostatnie zadanie będzie polegało tylko na przeniesieniu jednego krążka ($n = 1$), co przedstawia przypadek bazowy. W takiej sytuacji możliwe będzie bezpośrednie przeniesienie krążka bez konieczności użycia słupka tymczasowego.

Utwórz program służący do rozwiązania problemu wież Hanoi. Skorzystaj z funkcji rekurencyjnej pobierające cztery parametry: liczbę krążków przeznaczonych do przeniesienia; słupek, na którym początkowo znajdują się krążki; słupek, na który krążki mają zostać przeniesione; słupek używany w charakterze tymczasowego.

Program powinien wyświetlać dokładne informacje o ruchach, który należy wykonywać, aby przenosić krążki między słupkami. Na przykład aby przenieść stos trzech krążków ze słupka A na C, program powinień wyświetlać następująco serię ruchów:
`A->C` (to oznacza przeniesienie krążka ze słupka pierwszego za trzeci)
`A->B`
`B->C`
`A->C`
`B->A`
`B->C`
`A->C`

Zadanie 4 Czy funkcja `main()` może być wywoływana rekurencyjnie? Napisz program, w którym funkcja `main()` zawiera statyczną zmienną lokalną `count` (deklaracja poprzedzona słowem kluczowym `static`) o wartości początkowej równej 1. Przeprowadź postinkrementację zmiennej `count` i wydrukuj jej wartość, a następnie wywołaj funkcję `main()`. Uruchom program. Jaki jest wynik jego działania?

Zadanie 5 Na czy polega działanie przedstawionego tutaj programu. Co stanie sie po zamianie miejscami wierszy 9 i 10?

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int c;
5
6      // wpisz ciąg tekstowy i wprowadź Ctrl + Z
7      if ((c = getchar()) != EOF)
8      {
9          main();
10         printf("%c", c);
11     }
12
13     return 0;
14 }
```

Zadanie 6 Na czy polega działanie przedstawionego tutaj programu.

```
1  #include <stdio.h>
2  unsigned int f(unsigned int a, unsigned int b);
3  int main(void){
4      printf("Podaj␣dwie␣cakowite␣liczby␣dodatnie:␣");
5      unsigned int x;
6      unsigned int y;
7      scanf("%u%u", &x, &y);
8
9      printf("Wynik␣to␣%u\n", f(x,y));
10     return 0;
11 }
12 // parametr b musi być dodatni,
13 // aby uniknąć rekurencji nieskończonej
14 unsigned int f(unsigned int a, unsigned int b){
15     if (b ==1)
16         return a;
17     else
18         return a + f(a, b-1);
19 }
```

Po ustaleniu sposobu działania programu zmodyfikuj go w taki sposób, aby działał poprawnie po usunięciu ograniczenia polegającego na tym, że drugi argument funkcji `f` musi być nieujemny.

Zadanie 7 Funkcja Ackermanna jest wykorzystywana do sprawdzania, jak szybko komputer wykonuje operacje rekurencyjne. Napisz funkcję Ackermanna `A()` daną wzorem rekurencyjnym:

$$A(m,n) = \begin{cases} n+1, & \text{gdy } m = 0, \\ A(m-1,1), & \text{gdy } m > 0 \text{ i } n = 0 \\ A\Big(m-1, A(m,n-1)\Big), & \text{gdy } m > 0 \text{ i } n > 0. \end{cases}$$

Napisz program demonstrujący wywołanie funkcji z następującymi argumentami:

Uzupełni kod funkcji `A()` o odpowiednie wywołania funkcji `printf()` pozwalające na śledzenie wywołań rekurencyjnych.

Zadanie 8 Napisz program, w którym zdefiniowane są następujące funkcje:

- funkcja `create_array()` - tworzy dynamicznie tablicę dwuwymiarową o rozmiarach m na n liczb typu `int` o rozmiarze przekazanym przez argumenty wywołania i zwraca wskaźnik do zaalokowanego bloku pamięci;
- funkcja `complete_array()` - wypełnia tablicę dwuwymiarową przekazaną jej jako argument (tablica utworzona prze pierwszą funkcja) liczbami losowymi z przedziału od 0 do 1000, która jest jej pierwszym argumentem , trzecim i czwartym argumentem wywołania funkcji `complete_array()` powinny być wymiary tablicy;
- funkcja `print_array()` - wyświetla elementy tablicy dwuwymiarowej, będącej argumentem wywołania, drugim i trzecim argumentem tej funkcji powinny być wymiary tablicy;
- funkcja `print_sum_rows()` - oblicza i wyświetla sumę elementów każdego wiersza tablicy dwuwymiarowej, będącej argumentem wywołania, drugim i trzecim argumentem tej funkcji powinny być wymiary tablicy;
- funkcja `print_sum_columns()` - oblicza i wyświetla sumę elementów każdej kolumny tablicy dwuwymiarowej, będącej argumentem wywołania, drugim i trzecim argumentem tej funkcji powinny być wymiary tablicy.

Przetestuj działanie funkcji w programie dla wymiarów tablicy podanych przez użytkownika.

Zadanie 9 Napisz program, który tworzy plik tekstowy *numbers.txt*, który jest wypełniony liczbami w następujący sposób:

- wylosuj liczbę całkowitą n z przedziały od 50 do 500 i zapisz ją w pliku w pierwszym wierszu pliku;
- powtórz n razy następującą operację:
  - wylosuj liczbę całkowitą m z przedziały od 2 do 80 zapisz ją w pliku w osobnym wierszu:
  - wylosuj m liczb całkowitych z przedziału od -500 do 500 zapisz je w kolejnym wierszu oddzielone spacją.

Napisz funkcję/funkcje, która wykonuje powyższe zadanie/zadania.

Zadanie 10 Napisz program, który otwiera plik *numbers.txt* utworzony w zadaniu 9 do odczytu i na podstawie danych tam umieszczonych utworzy dynamicznie tablicę postrzępioną o n wierszach. Każdy z n wierszy ma mieć m kolumn wypełnionych odpowiednimi liczbami zapisanymi w pliku *numbers.txt* (n i m takie jak w zadaniu 9). Napisz funkcję/funkcje, która wykonuje powyższe zadanie/zadania.

Zadania dodatkowe:

Zadanie 1 Napisz iteracyjną wersję programu rozwiązującego problem wież Hanoi (zadanie 3).