

Lista zadań nr 3

Rekurencja

Zadanie 1.

Prześledź działanie podanych poniżej programów (A-E) ustalając jakie wartości i w jakiej kolejności zostaną wyświetlone w wyniku ich wykonania. Narysuj schemat wywołań rekurencyjnych, a następnie wykonaj każdy program. Czy wszystkie funkcje rekurencyjne zostały poprawnie zdefiniowane?

A.	<pre>void p(int x){ printf("%d\n",x); if (x<3) p(x+1); printf("%d\n",-x); } int main(void){ p(1); return 0; }</pre>
B.	<pre>void q(int x){ if (x<3) q(x+1); printf("%d\n",x); if (x<4) q(x+2); } int main(void){ q(1); return 0; }</pre>
C.	<pre>void q(int x){ if (x<3 && x>-4){ q(-x-1); printf("%d\n",2-x); } } int main(void){ q(0); return 0; }</pre>
D.	<pre>int f(int x){ printf("%d\n",x); if (x>1) return f(x/2)*3; return 1; } int main(void){ printf("%d",f(11)); return 0; }</pre>

E.	<pre> int f(int x){ printf("%d\n",x); if (x<3) return x; return f(x-1)+f(x-3); } int main(void){ printf("%d\n",f(5)); return 0; } </pre>
----	---

Zadanie 2.

Napisz funkcje rekurencyjne obliczające i zwracające dla zadanego argumentu n :

- sumę $1 + 3 + 5 + \dots + (2n-1)$
- iloczyn $1 * 4 * 7 * \dots * (3n-2)$

Na przykład dla $n=3$, funkcja sumująca powinna obliczyć i zwrócić $1+3+5$, czyli 9. Wykorzystaj obie funkcje w przykładowym programie.

Zadanie 3.

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

Zadanie 4.

Największy wspólny dzielnik liczb całkowitych x oraz y ($x \leq y$) 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. Wykorzystaj funkcję w przykładowym programie.

Zadanie 5.

Zdefiniuj funkcję rekurencyjną, która otrzyma jako argument liczbę całkowitą dodatnią, a wypisze jej reprezentację binarną. Wykorzystaj funkcję w przykładowym programie.

Zadanie 6.

Napisz rekurencyjną wersję funkcji `LinearSearch()` sprawdzającej czy w tablicy nieuporządkowanej znajduje się element o zadanej wartości (algorytm wyszukiwania liniowego). Tablica i szukana liczba są argumentami wywołania (być może również inne argumenty będą potrzebne), a funkcja zwraca indeks (pierwszego od końca tablicy) znalezionego elementu lub -1 w przypadku, gdy elementu nie ma. Wykorzystaj funkcję w przykładowym programie.

Wersja iteracyjna funkcji:

```
int LinearSearch(int t[], int n, int x)
{
    for(int i=0;i<n;i++)
        if ( t[i] == x)
            return i;
    return -1;
}
```

Zadanie 7.

Napisz rekurencyjną wersję funkcji `BinarySearch ()` binarnego wyszukiwania elementu w tablicy uporządkowanej. Wykorzystaj funkcję w przykładowym programie.

Wersja iteracyjna funkcji:

```
int BinarySearch(int t[], int n, int x){
    int left, right, mean;

    left =0;
    right=n-1;

    do{
        mean=(left+right) / 2;
        if ( t[mean] < x)
            left=mean+1;
        else
            right=mean-1;
    }while (t[mean]!=x && left<=right);

    if (t[mean]==x)
        return mean;
    else
        return -1;
}
```

Zadanie 8.

Zdefiniuj funkcję rekurencyjną, która odwróci kolejność elementów w tablicy. Wykorzystaj funkcję w przykładowym programie.

Zadanie 9.

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 post inkrementację zmiennej `count` i wydrukuj jej wartość, a następnie wywołaj funkcję `main()`. Uruchom program. Jaki jest wynik jego działania?

Zadanie 10.

Przeanalizuj działanie przedstawionego poniżej programu. Jaki będzie działał program po zamianie miejscami wierszy nr 9 i nr 10?

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

Zadanie 11

Funkcja Ackermanna może być wykorzystywana do sprawdzania, jak komputer wykonuje operacje rekurencyjne. Napisz funkcję Ackermanna `A()` daną wzorem rekurencyjnym:

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

Uzupełnij kod funkcji `A()` o odpowiednie wywołania funkcji `printf()`, pozwalające na śledzenie wartości argumentów wywołań rekurencyjnych. Napisz program demonstrujący wywołanie funkcji z następującymi argumentami:

`A(0, 0)`

`A(0, 1)`

`A(1, 1)`

`A(1, 2)`

`A(1, 3)`

`A(2, 2)`

`A(3, 2)`

Jak przebiegło wywołanie funkcji `A(4, 1)`?