szczególna klasa algorytmów, które znacząco zwiększają złożoność pamięciową. Są to algorytmy rekurencyjne. Każde wywołanie rekurencyjne powoduje umieszczenie na stosie adresu procedury wywołującej, a dodatkowo, jeżeli procedura rekurencyjna posiada zmienne lokalne lub parametry, są one także umieszczane na stosie przy każdym wykorzystaniu funkcji. Wielkość wykorzystanej pamięci zależy więc od maksymalnej głębokości rekurencji.

Przykład 12. Złożoność pamięciowa algorytmu rekurencyjnego obliczania silni

Dla liczby n głębokość rekurencji jest w algorytmie rekurencyjnego obliczania silni równa n. W języku Pascal każde wywołanie powoduje umieszczenie na stosie adresu funkcji wywołującej (4 bajty) oraz parametru n typu *word* (2 bajty). Złożoność pamięciowa algorytmu rekurencyjnego obliczania silni wynosi zatem $P(n) = (4 + 2) \cdot n = 6n$.

Ćwiczenie 9.

Otwórz plik *T7_c9.pas* lub *T7_c9.cpp* (CD). Dla jakiego rzędu wielkości parametru n wykonanie programu spowoduje błąd przepełnienia stosu (ang. *stack overflow error*)?

Złożoność obliczeniowa i pamięciowa algorytmów są ze sobą powiązane. Niekiedy można zmniejszyć złożoność obliczeniową algorytmu, np. dokonując tablicowania funkcji (wstępnego obliczenia wartości funkcji), co jednak zwiększa złożoność pamięciową algorytmu. Najczęściej prawdziwa jest również zależność odwrotna.

6. Efektywność algorytmów

Ze złożonością wiąże się pojęcie **algorytmu optymalnego**. Algorytm optymalny to taki, który dla rozwiązania danego problemu jest bezwzględnie najlepszy (czego można dowieść matematycznie). Złożoność obliczeniowa algorytmu jest pojęciem teoretycznym. W praktyce bardziej interesuje nas **efektywność algorytmu**, która bierze pod uwagę praktyczne zastosowanie algorytmu w programie.

Przykład 13.

Algorytm *A1* posiada złożoność obliczeniową $O(2n)$, a algorytm *A2* – złożoność obliczeniową $O(n)$. Algorytm *A1* działa jednak na liczbach całkowitych, a algorytm *A2* – na liczbach rzeczywistych. Ponieważ operacje na liczbach całkowitych są wykonywane przez procesor kilkakrotnie szybciej niż operacje na liczbach rzeczywistych, w praktyce algorytm *A1* będzie bardziej efektywny od algorytmu *A2*.

Ćwiczenie 10.

Algorytm *A1* posiada złożoność obliczeniową $O(2^{2n})$, a algorytm *A2* – złożoność obliczeniową $O(n!)$. Dla jakich wartości n algorytm *A1* jest bardziej efektywny od algorytmu *A2*?

Wskazówka: Wykonaj odpowiednie obliczenia w arkuszu kalkulacyjnym.
Mimo wzrostu mocy obliczeniowej komputerów, pewne klasy algorytmów (np. tzw. problem komiwojażera, łamanie haseł zakodowanych najnowszymi metodami szyfrowania) dla odpowiednio dużych zbiorów danych wciąż pozostają poza zasięgiem ich możliwości.