Wielkopolska Wyższa Szkoła Społeczno-Ekonomiczna
w Środzie Wielkopolskiej

LAB 2

Algorytmy i struktury danych – rekurencja vs iteracja - zastosowania.

1. Opracuj algorytm sprawdzania, czy wprowadzona przez użytkownika dowolna liczba całkowita jest dodatnia: wyprowadź wówczas, jako wynik napis **dodatnia** lub (**+**). Jeśli liczba ta jest zerem, podaj jako wynik napis **zero** lub wartość **0**. Jeśli liczba ta jest ujemna, wyprowadź napis **ujemna** lub (**-**). Ponadto na końcu algorytmu użytkownik powinien być zapytany czy chce kontynuować, czy zakończyć działanie programu np. Czy chcesz kontynuować? [T/N]

2. Przedstaw algorytm obliczania tygodniowego zarobku pracownika na podstawie liczby przepracowanych godzin. Wynagrodzenie podstawowe **PP** jest iloczynem liczby godzin **LG** i stawki godzinowej **SG**. Dla liczby godzin powyżej **42** należy doliczyć wynagrodzenie dodatkowe **NG** zakładając współczynnik **2**. W wyniku podać liczbę godzin przepracowanych **LG** oraz obliczoną płacę **PL**.
3. Dodatkowo dla obliczonej pensji proszę wyliczyć roczne wynagrodzenie i podatek.
   a) Zarobki w kwocie do **20.000** podlegają stawce podatkowej w wysokości **19%**.
   b) Zarobki w kwocie od **20.000** do **30.000** podlegają stawce podatkowej w wysokości **21%**.
   c) Zarobki w kwocie powyżej **30.000** podlegają stawce podatkowej w wysokości **28%**.
   d) W wyniku podaj roczne zarobki, stawkę podatkową, obliczony podatek i wynagrodzenie po potrąceniu podatku.
4. *Przedstaw algorytm obliczania **n!** metodą iteracyjną z wykorzystaniem pętli for oraz z wykorzystaniem rekurencji, zgodnie ze wzorem: **n! = n (n – 1)!** dla **n >=1**, **0! = 1**.

Powyższe algorytmy zaimplementuj algorytm w języku C++.

Każdy program powinien:
⇨ „przedstawić" się,
⇨ pytać użytkownika o kontynuację (T/N),
⇨ posiadać proste zabezpieczenia (IF itp.) przed błędnym wprowadzaniem danych.

POMOCE

Rekurencją nazywamy wywoływanie funkcji wewnątrz jej definicji.

Iteracja – czynność powtarzania (najczęściej wielokrotnie) tej samej instrukcji (albo wielu instrukcji) w pętli. Mianem iteracji określa się także operacje wykonywane wewnątrz takiej pętli.

Silnia iteracyjnie

Przykład 1a

```
#include <cstdlib>
#include <iostream>
using namespace std;
//-----funkcja silnia iteracyjnie--------
int main(int argc, char *argv[])
{
    int n,i,silnia = 1;
    cout <<"================================================\n";
    cout <<"                 Silnia iteracyjnie             \n";
    cout <<"                                                \n";
    cout <<"                 WWSSE(C) 2019 student          \n";
    cout <<"================================================\n\n";
    cout<< "Podaj liczbe naturalna n!= ";
    cin>>n;
    for(i=1; i<=n; i++)
    {
    silnia *= i;
    }
    cout<<endl<<"Silnia liczby "<< n <<" wynosi: "<<silnia<<endl<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Przykład 1b

```
#include <cstdlib>
#include <iostream>
using namespace std;
//-----funkcja silnia iteracyjnie--------
int silnia (int n)
    {
    int i, silnia = 1;
    for(i=1; i<=n; i++)
    {
    silnia *= i;
    }
return silnia;
}
//-------------koniec funkcji--------------
int main(int argc, char *argv[])
{
    int n;
    cout <<"================================================\n";
    cout <<"                 Silnia iteracyjnie             \n";
    cout <<"                                                \n";
    cout <<"                 WWSSE(C) 2019 student          \n";
    cout <<"================================================\n\n";
    cout<< "Podaj liczbe naturalna n!= ";
    cin>>n;
    cout<<endl<<"silnia liczby "<< n <<" wynosi: "<<silnia(n)<<endl<<endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Silnia rekurencyjnie

Przykład 2

```
#include <cstdlib>
#include <iostream>
using namespace std;
//-----funkcja silnia rekurencyjnie--------
int silnia (int n)
    {
    if (n == 0)
    return 1;
    else
    return n*silnia(n - 1);
    }
//-------------koniec funkcji--------------
int main(int argc, char *argv[])
{
    int n;
    cout <<"================================================\n";
    cout <<"              Silnia rekurencyjnie              \n";
    cout <<"                                                \n";
    cout <<"                WWSSE(C) 2019 student             \n";
    cout <<"================================================\n\n";
    cout<< "Podaj liczbe naturalna n!= ";
    cin>>n;
    cout<<endl<<"Silnia liczby "<< n <<" wynosi: "<<silnia(n)<<endl<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Kontynuacja działania algorytmu w petli bądź jego przerwanie – zastosowanie pętli **do while**.

Przykład 3

```
#include <cstdlib>
#include <iostream>
using namespace std;
void baner()
{
    cout <<"|=========================================|\n"
         <<"|        Program oblicza pole kwadratu    |\n"
         <<"|            WWSSE(C)2019 student         |\n"
         <<"|=========================================|\n\n";
}
int main(int argc, char *argv[])
{
     int boka, pole;
     char znak;
     baner();
do
    {
     cout<<"Podaj dlugosc boku kwadratu a = ";
     cin>>boka;
     pole=boka*boka;
     cout<<"\n\nPole kwadratu wynosi: "<<pole<<endl<<endl;
     }
     cout << "Czy chcesz policzyć jeszcze raz? [T/N] ";
     cin >> znak;
     cout <<endl;
}
    while (znak=='T' || znak=='t');
    system("PAUSE");
    return EXIT_SUCCESS;
}
```