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. Opracuj algorytm pobierający od użytkownika **trzy** liczby i wyświetlający je na ekranie w porządku rosnącym, bez używania tablic.

3. Napisz program wyświetlający słowną reprezentację liczb od **0** do **10**. Jeśli użytkownik wprowadzi np. **3** program wyświetli napis **trzy**☺.

4. Napisz program 'Czy liczba jest parzysta' pobierający od użytkownika jedną liczbę i drukujący na ekranie jedną z informacji **LICZBA JEST PARZYSTA** bądź **LICZBA JEST NIEPARZYSTA**. Proszę zrealizować to zagadnienie na trzy sposoby (modulo, logicznie i…).

5. Napisz program **Potęga n-ta** pobierający od użytkownika dwie liczby: **x** (rzeczywiste – podstawa potęgi) oraz **n** (naturalne z dowolnego zakresu – wykładnik potęgi).

6. Program ma drukować na ekranie liczbę $\mathbf{x^n}$. Proszę zrealizować to zagadnienie na dwa sposoby (pętla i funkcja **POW**).

7. 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**.
8. 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.

9. *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

Instrukcja warunkowa if/else

Instrukcja warunkowa umożliwia wykonanie pewnej instrukcji w zależności od wartości wyrażenia. Wszystkie wartości różne od 0 są w języku C traktowane jako prawda, równe 0 jako fałsz. Wyrażenia logiczne są liczone tylko do momentu, w którym można określić jego wartość.

.....:::::Instrukcja pojedynczego wyboru (if):::::.....

```
if (wyrażenie)
instrukcja;
```

.....:::::Instrukcja podwójnego wyboru (if/else):::::.....

```
if (wyrażenie)
instrukcja1;
else instrukcja2;
```

W obu rozkazach instrukcja może być instrukcją złożoną. W pierwszym przypadku instrukcja wykonuje się, jeśli wartość wyrażenia jest różna od 0. W drugim wykonuje się jedna z dwóch podanych instrukcji (nigdy obie) - pierwsza, gdy wartość wyrażenia jest różna od 0, druga - gdy wartość wyrażenia jest równa 0.

Przykład:

a)

```
if (a > 5)
cout << "a jest wieksze od 5\n";
```

```
else
cout << "a jest mniejsze lub rowne 5\n";
```

b)

```
cout << "Program dla doroslych\n Ile Ty masz lat? \n";
cin >> wiek;
if (wiek < 18) cout << "Jesteś niepełnoletni! \n";
else cout << "Możesz wejść dalej\n";
```

Jeśli po spełnieniu lub niespełnieniu warunku ma się wykonać więcej niż jedna instrukcja, trzeba taki ciąg instrukcji zamknąć w klamrach. Kilka instrukcji zamknięte w nawiasy klamrowe traktowane są przez kompilator jako jedna instrukcja.

Przykład:

```
if (warunek)
{
 instrukcja1;
 instrukcja2;
 instrukcja3;
}
else instrukcja4;
```

W takim przypadku nie daje się już średnika przed **else**, bo zamykający nawias klamrowy pełni tutaj podobną rolę jak średnik. W taki sam sposób można umieścić blok instrukcji po słówku **else**.

Instrukcja wielokrotnego wyboru switch

```
switch (warunek)
{
 case 1: instrukcja1; break;
 case 5: instrukcja2; break;
 case 8: instrukcja3; break;
 default: instrukcja4;
}
```

Gdy zmienna **warunek** będzie miała wartość **1**, wykonają się wszystkie instrukcje od dwukropka do najbliższej instrukcji **break** [czyli **instrukcja1**]. Gdy zmienna **warunek** będzie miała wartość **5**, wykona się **instrukcja2**. No i wreszcie gdy zmienna **warunek** będzie miała wartość **8**, wykona się **instrukcja3**. Jeśli zmienna **warunek** nie będzie równa żadnej z wymienionych wartości, wykonają się instrukcje "awaryjne" umieszczone po słówku **default**. Jeśli zapomnimy słówka **break**, program będzie wykonywał wszystkie instrukcje po kolei dopóki nie napotka instrukcji **break** albo dopóki nie skończy się instrukcja warunkowa. Zazwyczaj jest to efekt niepożądany.

Przykład 1

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
int boka, wybor, pole, obwod;

      cout <<"Program oblicza pole i obwod kwadratu\n";
      cout <<"Wybierz co chcesz obliczyc:\n";
      cout <<"1 -> Pole\n";
      cout <<"2 -> Obwod\n";
      cin >> wybor;
      switch (wybor)
      {
      case 1:
            cout <<"Podaj dlugosc boku kwadratu a="; cin>>boka;
            pole=boka*boka;
            cout <<"Pole kwadratu wynosi: "<<pole<<endl;
            break;
      case 2:
            cout <<"Podaj dlugosc boku kwadratu a="; cin>>boka;
            obwod=4*boka;
            cout <<"Obwod kwadratu wynosi: "<<obwod<<endl;
            break;
      default:
            cout <<"Prosze wybrac jedna z opcji 1 lub 2";
      }
    system("PAUSE");
   return EXIT_SUCCESS;
}
```

Parzystość liczby

bitowo

```
if (liczba & 1)
      cout << "Liczba jest nieparzysta \n";
else
      cout << "Liczba jest parzysta \n";
```

modulo

```
if (liczba %2 ==0)
      cout << "Liczba jest parzysta \n";
else
      cout << "Liczba jest nieparzysta \n";
```

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;
}
```