**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 1\_03 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Obliczanie wartość wielomianu 2 stopnia | | |

1. Treść

Zaimplementuj algorytm obliczający wartość wielomianu 2 stopnia metodami klasyczną i Hornera

* 1. Metoda realizacji

Po wprowadzeniu z klawiatury czynników wielomianu obliczamy wartość wielomianu metodą klasyczną i Hornera

* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe *(opis)*

a0,a1,a2 – czynniki wielomianu – wprowadzane z klawiatury

n – wartość potęgi wielomianu – wprowadzana z klawiatury

* + 1. Dane wyjściowe *(opis)*

Wartość wielomianu – liczba całkowita – wprowadzana na ekran

1. Realizacja
   1. Metoda klasyczna
      1. Algorytm *(lista kroków lub schemat blokowy)*

a)Deklarujemy zmienną pomocniczą o wartości 0

b) Dodajemy do zmiennej pomocniczej a0

c) Dodajemy do zmiennej pomocniczej a1 \*n

d) Dodajemy do zmiennej pomocniczej a2 \*n^2

e) Zwracamy wynik

Kod źródłowy

int wielomian\_klasycznie(int a2,int a1,int a0,int n)

{

int pom=0;

pom=pom+a0;

pom=pom+a1\*n;

pom=pom+a2\*n\*n;

return pom;

}

int wielomian\_horner(int a2,int a1,int a0,int n)

{

int pom;

pom=a2\*n+a1;

pom=pom\*n+a0;

return pom;

}

int main()

{

int a2,a1,a0,n;

cin >>a2>>a1>>a0>>n;

cout<<wielomian\_klasycznie(a2,a1,a0,n)<<endl;

cout<<wielomian\_horner(a2,a1,a0,n)<<endl;

return 0;

}

#include <iostream>

using namespace std;

int fibbo\_iteracyjnie(int n)

{

if(n<=2)return 1;

int pom;

int f1=1;

int f2=1;

for(int i=3;i<=n;i++)

{

pom=f1+f2;

f1=f2;

f2=pom;

}

return pom;

}

int main()

{

int n;

cin>>n;

cout<<fibbo\_iteracyjnie(n)<<endl;

return 0;

}

* + 1. Dane wejściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

* + 1. Dane wyjściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

* 1. Schemat Hornera
     1. Algorytm *(lista kroków lub schemat blokowy)*

a)Deklarujemy zmienną pomocniczą o wartości a2\*n+a1

b)Mnożymy zmienną pomocniczą przez n

c)Dodajemy do zmiennej pomocniczej a0

* + 1. Kod źródłowy

#include <iostream>

using namespace std;

int fibbo\_rekurencyjnie(int n)

{

if(n==0)return 0;

if(n==1)return 1;

return fibbo\_rekurencyjnie(n-1)+fibbo\_rekurencyjnie(n-2);

}

int main()

{

int n;

cin>>n;

cout<<fibbo\_rekurencyjnie(n)<<endl;

return 0;

}

* + 1. Dane wejściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

* + 1. Dane wyjściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

1. Wnioski *(Określ złożoność asymptotyczną algorytmów przy użyciu notacji* ***O lub innej i inne wnioski)***

Dla rozwiązania klasycznego: O(1), ponieważ funkcja posiada stały czas obliczeniowy

Dla rozwiązania Hornera: O(1), ponieważ funkcja posiada stały czas obliczeniowy