Prowadzący zajęcia  
dr inż. Andrzej Urbański

**Laboratoria nr2**

**Programowanie Niskopoziomowe**

**Tablice i pętle w C**

Wojciech Regulski  
Informatyka(WI) I1  
nr 132312

Zadanie nr 1.

Odwracanie tablicy jednowymiarowej.

int tab[n], i, temp;

for(i=0; i<n; i++)

{

tab[i]=i;

printf("%d\n",tab[i]);

}

for(i=0; i<n/2; i++)

{

temp = tab[n-i-1];

tab[n-i-1]=tab[i];

tab[i]=temp;

}

printf("\nOdwrocona:\n");

for(i=0; i<n; i++)

{

printf("%d\n",tab[i]);

}

Zamieniana jest wartość pierwszego elementu z ostatnim, drugiego z przedostatnim itd.,   
aż do elementu środkowego.

Zadanie nr 2.

Znajdowanie największej wartości wyrazu ciągu Fibonacciego nie większej od danej liczby podanej z wejścia (iteracyjnie).

int in, stare=1, nowe=1,temp;

scanf("%d",&in);

if(in<=1)

{

printf("1");

return 0;

}

while(1)

{

temp=nowe;

nowe=stare+nowe;

stare=temp;

if(stare<=in && nowe>in)

{

printf("%d",stare);

break;

}

}

Każdy wyraz ciągu bazuje na dwóch poprzednich, potrzebne są więc 3 zmienne. Gdy nowoutworzona jest już większa od wejściowej liczby, to wypisana zostaje poprzednia – ostania nie większa od podanej.Zadanie nr 3.

Obliczanie NWD dla 2 wejściowych liczb naturalnych.

int a,b;

scanf("%d",&a);

scanf("%d",&b);

while(a!=b)

{

if(a>b)

a-=b;

else

b-=a;

}

printf("%d",a);

Implementacja prostego algorytmu.

Zadanie nr 4.

Obliczanie NWW dla 2 wejściowych liczb naturalnych.

int a,b,c,d;

scanf("%d",&a);

scanf("%d",&b);

c=a; d=b;

while(a!=b)

{

if(a>b)

a-=b;

else

b-=a;

}

printf("%d",c\*d/a);

Implementacja prostego algorytmu.Zadanie nr 5.

Liczba doskonała to liczba równa sumie swoich podzielników mniejszych od niej samej. Napisz program znajdujący wszystkie liczby doskonałe w przedziale <1, a>, np.: 28 = 1 + 2 + 4 + 7 + 14.

int sum\_dziel(int liczba)

{

int i, suma=0;

for(i=1;i<=liczba/2;i++)

{

if(liczba%i==0)

suma+=i;

}

return suma;

}

int main()

{

int a,i;

scanf("%d",&a);

for(i=1;i<=a;i++)

{

if(i==sum\_dziel(i))

printf("%d\n",i);

}

return 0;

}

Funkcja sum\_dziel zwraca sumę dzielników podanej liczby przechodząc przez wszystkie mniejsze/równe od jej połowy. Jeśli suma jest równa tej liczbie, to wypisuje ją na ekran.Zadanie nr 6.

Liczby zaprzyjaźnione to takie dwie liczby, z których każda jest równa sumie podzielników drugiej liczby mniejszych od tej liczby. Na przykład 220 i 284 są zaprzyjaźnione, ponieważ: 220 dzieli się przez 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 i 110, których suma wynosi 284, zaś 284 dzieli się przez 1, 2,

71 i 142, których suma wynosi 220.

Napisz program znajdujący wszystkie liczby zaprzyjaźnione z przedziału [1, a].

int sum\_dziel(int liczba)

{

int i, suma=0;

for(i=1;i<=liczba/2;i++)

{

if(liczba%i==0)

suma+=i;

}

return suma;

}

int main()

{

int a,i; scanf("%d",&a);

int tab[a+1];

for(i=1;i<=a;i++)

{

tab[i]=sum\_dziel(i);

}

for(i=1;i<=a;i++)

{

if(tab[i]<a)

if(tab[tab[i]]==i && tab[i]>i)

printf("[ %d %d ]\n",i,tab[i]);

}

return 0;

}

Obliczone zostają sumy dzielników, po czym sprawdzane zostaje, czy suma dzielników danej liczby ma sumę dzielników równą tej liczbie.

Zadanie nr 7.

Liczba pierwsza N to taka, która dzieli się tylko przez 1 i N. Wyznaczenie wszystkich liczb pierwszych w przedziale <A, B> algorytmem sita Erastotenesa.

int i,j,zakres,dokad;

scanf("%d",&zakres);

int tablica[zakres];

dokad=floor(sqrt(zakres));

for (i=1; i<=zakres; i++) tablica[i]=1;

for (i=2; i<=dokad; i++)

{

if (tablica[i] != 0)

{

j = i+i;

while (j<=zakres)

{

tablica[j] = 0;

j += i;

}

}

}

for (i=2; i<=zakres; i++)

if (tablica[i]!=0)

printf("%d, ",i);

Na początku wszystkie liczby są oznaczane jako pierwsze, po czym wielokrotności liczb pierwszych są odznaczane. Po całym przejściu oznaczone jako pierwsze są tylko poprawne liczby.Zadanie nr 8.

Sprawdzić prawdziwość twierdzenia, że każda liczba parzysta jest sumą dwóch liczb pierwszych.

int l\_pierwsza(int n){

int i;

for(i=2; i<=n/2; i++) {

if (n%i == 0) {

return 0;

}

}

return 1;

}

int main()

{

int zakres=1000, i, j, k, done=0;

for(i=2; i<=zakres; i+=2)

{

for(k=1; k<i; k++)

if(done == 0)

{

for(j=k+1; j<i; j++)

{

if(k+j==i && l\_pierwsza(k) && l\_pierwsza(j))

{

printf("Liczba: %d = %d + %d (twierdzenie prawdziwe)\n",i, k, j);

done=1;

}

}

}

done=0;

}

return 0;

}

Przechodząc przez kolejne liczby zostaje sprawdzone, czy ich suma jest pożądaną liczbą, jeśli tak czy suma składa się z liczb pierwszych.Zadanie nr 9.

Liczby bliźniacze to liczby pierwsze, których różnica wynosi dwa, a więc np.: 3-5, 11-13.Znajdź wszystkie pary liczb bliźniaczych z przedziału <1, a>.

int i,j,zakres,dokad;

scanf("%d",&zakres);

int tablica[zakres];

dokad=floor(sqrt(zakres));

for (i=1; i<=zakres; i++) tablica[i]=1;

for (i=2; i<=dokad; i++)

{

if (tablica[i] != 0)

{

j = i+i;

while (j<=zakres)

{

tablica[j] = 0;

j += i;

}

}

}

for (i=4; i<=zakres; i++)

if (tablica[i]!=0 && tablica[i-2]!=0)

printf("[%d %d], ",i-2,i);

Znalezienie liczb pierwszych sitem, po czym przeiterowanie po kolejnych liczbach w poszukiwaniu liczb pierwszych oddanych od siebie o 2.

Zadanie nr 10.

Liczba Mersenne'a to liczba pierwsza postaci 2p-1, przy czym p samo jest liczbą pierwszą. Napisz program znajdujący takie liczby w przedziale <1, a>.

int sum\_dziel(int liczba)

{

int i, suma=0;

for(i=1;i<=liczba/2;i++)

{

if(liczba%i==0)

suma+=i;

}

return suma;

}

int main()

{

int p=2,a;

scanf("%d",&a);

while(pow(2,p)<=a)

{

if(sum\_dziel(p)==1)

if(sum\_dziel(pow(2,p)-1)==1)

printf("%.0f\n",pow(2,p)-1);

p++;

}

return 0;

}

Przeiterowanie po kolejnych liczbach i sprawdzenie żądanego warunku.Zadanie nr 11.

Każda liczba całkowita może być jednoznacznie rozłożona na iloczyn potęg liczb pierwszych. Napisz program, który dla danej liczby całkowitej dokonuje rozkładu.

int n, a, i;

printf("Podaj liczbe: ");

scanf("%d",&a);

n=2;

i=0;

while(a>1)

{

while(a%n==0)

{

i++;

a/=n;

}

if(i!=0)printf("%d ^ %d\n",n, i);

i=0;

n++;

}

return 0;Zadanie nr 12.

Napisać funkcję potęgowania liczby naturalnej za pomocą mechanizmu wielokrotnego dodawania(iteracyjnie).

int potegowanie(int podstawa, int wykladnik)

{

if(wykladnik==0) return 1;

int wynik=podstawa,i,j;

for(i=1;i<wykladnik;i++)

{

int temp=wynik;

for(j=1;j<podstawa;j++)

{

wynik+=temp;

}

}

return wynik;

} Zadanie nr 13.

Napisz program wyznaczający resztę z dzielenia liczby naturalnej z wykorzystaniem mechanizmu

wielokrotnego odejmowania.

int liczba=29;

int dzielnik=6;

while(liczba>=0)

liczba-=dzielnik;

liczba+=dzielnik;

printf("%d",liczba); Zadanie nr 14.

Napisać funkcję wyznaczania silni liczby naturalnej n (iteracyjnie).

int silnia(int liczba)

{

if(liczba<2) return 1;

int i;

for(i=liczba-1;i>1;i--)

{

liczba\*=i;

}

return liczba;

} Zadanie nr 15.

Hipoteza Simmonsa mówi, że tylko 4 silnie można wyrazić jako iloczyny trzech kolejnych liczb całkowitych. Oto jedna z nich: 4! = 2 \* 3 \* 4. Znajdź trzy pozostałe. Czy możesz ich znaleźć więcej i obalić hipotezę?

long long silnia(long long liczba)

{

if(liczba<2) return 1;

int i;

for(i=liczba-1;i>1;i--)

{

liczba\*=i;

}

return liczba;

}

int main()

{

long long n=20,wynik=0;

long long tab[n],i,j;

for(i=0;i<n;i++)

tab[i]=silnia(i);

for(i=1;wynik<tab[n-1];i++)

{

wynik=i\*(i+1)\*(i+2);

for(j=0;j<n;j++)

if(wynik==tab[j])

printf("%I64d \* %I64d \* %I64d = %I64d!\n",i,i+1,i+2,j);

}

return 0;

}

Zostają znalezione 4, nie można sprawdzać dalej niż do 20!, bo zmienne nie są w stanie przechować tak wielkich liczb;

Zadanie nr 16.

Obliczyć ciąg symboli Newtona dla i = 1...k, (n po i) = n!/(i!\*(n -i)!) dla podanych liczb n, k.

long long silnia(long long liczba)

{

if(liczba<2) return 1;

int i;

for(i=liczba-1;i>1;i--)

{

liczba\*=i;

}

return liczba;

}

long long newton(long long n, long long k)

{

return silnia(n)/(silnia(k)\*silnia(n-k));

}

int main()

{

long long n=6, k=4;

long long i=1;

for(;i<=k;i++)

{

printf("%I64d po %I64d = %I64d\n",n,i,newton(n,i));

}

return 0;

}

Zadanie nr 17.

Napisz program wyszukiwania w podanym przedziale <D, G> wszystkich liczb naturalnych (pitagoryjskich) a, b, dla których istnieje liczba naturalna c spełniająca warunek a2+ b2= c2.

int d, g;

int a, b;

float c;

printf("Podaj zakres szukania: ");

scanf("%d %d", &d, &g);

for(a = d; a < g; a++)

for(b = a; b < g; b++)

if((c = sqrt(pow(a, 2) + pow(b, 2))) == roundf(c))

printf("a = %d \tb = %d\tc = %.0f\n", a, b, c);

return 0;

Zadanie nr 18.

Dla danej macierzy wyznaczyć sumę liczb na głównej przekątnej.

srand(time(NULL));

int matrix\_size;

int i, j;

int max = 10;

int suma=0;

printf("Podaj rozmiar macierzy: ");

scanf("%d", &matrix\_size);

int matrix[matrix\_size][matrix\_size];

for(i = 0; i < matrix\_size; i++)

for(j = 0; j < matrix\_size; j++)

{

matrix[i][j] = rand() % (max+1);

if(i == j)

suma += matrix[i][j];

}

printf("Utworzona macierz:\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", matrix[i][j]);

}

printf("\n\n");

}

printf("Suma: %d\n", suma);

return 0;

Zadanie nr 19.

Dla danej macierzy wyznaczyć stosunek sumy wartości liczb nad główną przekątną do sumy wartości liczb pod główną przekątną.

srand(time(NULL));

int matrix\_size;

int i, j;

int max = 10;

int suma\_nad=0, suma\_pod=0;;

printf("Podaj rozmiar macierzy: ");

scanf("%d", &matrix\_size);

int matrix[matrix\_size][matrix\_size];

for(i = 0; i < matrix\_size; i++)

for(j = 0; j < matrix\_size; j++)

{

matrix[i][j] = rand() % (max+1);

if(i > j)

suma\_nad += matrix[i][j];

else if(j > i)

suma\_pod += matrix[i][j];

}

printf("Utworzona macierz:\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", matrix[i][j]);

}

printf("\n\n");

}

printf("Suma pod przekatna: %d\n", suma\_pod);

printf("Suma nad przekatna: %d\n", suma\_nad);

printf("Stosunek sumy wartosci liczb nad glowna przekatna do sumy wartosci liczb pod glowna przekatna: %f\n", (float)suma\_nad/suma\_pod);

return 0;

Zadanie nr 20.

Mnożenie dwóch macierzy.

srand(time(NULL));

int matrix\_size;

int i, j, k;

int max = 10;

int suma=0;

printf("Podaj rozmiar macierzy: ");

scanf("%d", &matrix\_size);

int matrix1[matrix\_size][matrix\_size];

int matrix2[matrix\_size][matrix\_size];

int multiplied\_matrix[matrix\_size][matrix\_size];

for(i = 0; i < matrix\_size; i++)

for(j = 0; j < matrix\_size; j++)

{

matrix1[i][j] = rand() % (max+1);

matrix2[i][j] = rand() % (max+1);

}

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

for(k = 0; k < matrix\_size; k++)

suma += matrix1[i][k] \* matrix2[k][j];

multiplied\_matrix[i][j] = suma;

suma = 0;

}

}

printf("\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", matrix1[i][j]);

}

printf("\n");

}

printf("\n");

for(i=0;i<matrix\_size/2;i++)

printf("\t");

printf("\*\n\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", matrix2[i][j]);

}

printf("\n");

}

printf("\n");

for(i=0;i<matrix\_size/2;i++)

printf("\t");

printf("=\n\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", multiplied\_matrix[i][j]);

}

printf("\n");

}

return 0;

Zadanie nr 21.

Napisz program czytania n i rysowania macierzy o rozmiarach n x n typu lewa-dolna/prawa-górna. Macierz lewa-dolna zawiera na przekątnej głównej i pod tą przekątną znaki X, natomiast powyżej przekątnej głównej są znaki spacji.

srand(time(NULL));

int matrix\_size;

int i, j;

printf("Podaj rozmiar macierzy: ");

scanf("%d", &matrix\_size);

printf("\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

if(j <= i)

printf("x");

else

printf(" ");

}

printf("\n");

}

printf("\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

if(j >= i)

printf("x");

else

printf(" ");

}

printf("\n");

}

return 0;

Zadanie nr 22.

Dla danej macierzy wejściowej napisać program sprawdzający czy jest ona symetryczna, czy zawiera

puste kolumny (tzn. wypełnione zerami).

srand(time(NULL));

int matrix\_size, i, j;

int max = 1;

printf("Podaj rozmiar macierzy: ");

scanf("%d", &matrix\_size);

int matrix[matrix\_size][matrix\_size];

for(i = 0; i < matrix\_size; i++)

for(j = 0; j < matrix\_size; j++)

matrix[i][j] = rand() % (max+1);

int is\_symmetric = 1;

for(i = 0; i < matrix\_size; i++)

for(j = 0; j < matrix\_size; j++)

if(matrix[i][j] != matrix[j][i])

{

is\_symmetric = 0;

break;

}

if(is\_symmetric == 0)

printf("Macierz nie jest symetryczna.\n");

else

printf("Macierz jest symetrycna.\n");

int contains\_empty\_columns;

for(j = 0; j < matrix\_size; j++)

{

contains\_empty\_columns = 1;

for(i = 0; i < matrix\_size; i++)

{

if(matrix[i][j] != 0)

{

contains\_empty\_columns = 0;

break;

}

}

if(contains\_empty\_columns == 1)

break;

}

if(contains\_empty\_columns == 0)

printf("Nie zawiera pustych kolumn.\n");

else

printf("Zawiera puste kolumny.\n");

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

Zadanie nr 23.

Dla danej macierzy wejściowej wyznaczyć stosunek sum wartości dwóch kolumn lub wierszy (decyduje użytkownik), przy czym numery tych kolumn/wierszy są również danymi wejściowymi.

srand(time(NULL));

int matrix\_size, i, j;

int max = 10;

double proportion;

char row\_or\_col;

int sel1, sel2, sum1=0, sum2=0;

printf("Podaj rozmiar macierzy: ");

scanf("%d", &matrix\_size);

int matrix[matrix\_size][matrix\_size];

for(i = 0; i < matrix\_size; i++)

for(j = 0; j < matrix\_size; j++)

matrix[i][j] = rand() % (max+1);

for(i = 0; i < matrix\_size; i++)

{

for(j = 0; j < matrix\_size; j++)

{

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

printf("Wpisz w jesli chcesz obliczac wiersze a k jesli kolumny, oraz ich numery.\n");

scanf(" %c", &row\_or\_col);

scanf("%d", &sel1);

scanf("%d", &sel2);

printf("\n");

sum1 = sum2 = 0;

if(row\_or\_col == 'w')

{

for(i = 0; i < matrix\_size; i++)

sum1 += matrix[sel1][i];

for(i = 0; i < matrix\_size; i++)

sum2 += matrix[sel2][i];

}

if(row\_or\_col == 'k')

{

for(i = 0; i < matrix\_size; i++)

sum1 += matrix[i][sel1];

for(i = 0; i < matrix\_size; i++)

sum2 += matrix[i][sel2];

}

proportion = (double) sum1 / sum2;

printf("sum1: %d, sum2: %d, prop: %lf\n", sum1, sum2, proportion);

Zadanie nr 24.

Zliczyć ilość wystąpień wzorca w ciągu bazowym (oba są ciągami znakowymi).

char\* baza = "alabbfdcalaigigiala";

char\* wzorzec = "ala";

int i, j, wystapienia=0;

char pierwszy\_znak = wzorzec[0];

for(i = 0; i < strlen(baza); i++)

{

if(baza[i] == pierwszy\_znak)

{

if( (i+strlen(wzorzec)) <= strlen(baza) )

for(j = 0; j < strlen(wzorzec); j++)

{

if(baza[i+j] != wzorzec[j])

break;

if(wzorzec[j+1] == '\0')

wystapienia++;

}

}

}

printf("Wystapienia wzorca: %d\n", wystapienia);

Zadanie nr 25.

Wypisać znak, który występuje największą ilość razy w znakowym ciągu wejściowym.

int liczba\_znakow = 26;

char alfabet[liczba\_znakow];

int min=97, max=122; //ascii 97-a 122-z

char \*input\_string = "co tam panie w polityce?";

int input\_strlen = strlen(input\_string);

int i;

for(i = 0; i < liczba\_znakow; i++)

{

alfabet[i] = 0;

}

for(i = 0; i < input\_strlen; i++)

{

char c = input\_string[i];

if(c>=min && c<=max)

alfabet[(int)(c - min)] += 1;

}

for(i = 0; i < liczba\_znakow; i++)

printf("%c pojawia sie: %d razy\n", 'a' + i, alfabet[i]);

Zadanie nr 26.

Wyznaczyć najdłuższy podciąg, wejściowego ciągu znakowego składający się z tych samych znaków

zdefiniowanych na wejściu.

char\* input\_string = "ddffffffcccc";

char l\_char, c\_char;

int l\_ilosc, c\_ilosc, i, j;

l\_char = '\0';

l\_ilosc = 0;

for(i = 0; i < strlen(input\_string); i++)

{

c\_char = input\_string[i];

c\_ilosc = 1;

while(c\_char == input\_string[i+1])

{

c\_ilosc += 1;

i++;

if(i == '\0')

break;

}

if(c\_ilosc > l\_ilosc)

{

l\_char = c\_char;

l\_ilosc = c\_ilosc;

}

}

printf("najdluzszy podciag jednego znaku sklada sie z '%c' i ma dlugosc: %d\n", l\_char, l\_ilosc);

Zadanie nr 27.

Sortowanie bąbelkowe

void BubbleSort(int tab[], int n)

{

int i,j;

for (i = 0; i < n - 1; i++)

{

for (j = 1; j < n - i; j++) //-i aby nie sortowac posortowanej czesci

{

if (tab[j - 1] > tab[j]) //zamien jesli poprzedni jest mniejszy

{

int temp = tab[j - 1];

tab[j - 1] = tab[j];

tab[j] = temp;

}

}

}

}

int main()

{

int n=10;

int \*tab;

tab = (int\*) malloc (sizeof (int) \* n);

int i;

for(i=0;i<n;i++)

{

tab[i]=rand()%100;

}

BubbleSort(tab,n);

for(i=0;i<n;i++)

{

printf("%d\n",tab[i]);

}

free(tab);

return 0;

}

Zadanie nr 28.

Sortowanie przez wybieranie

void SelectionSort(int tab[], int n)

{

int i,j;

for (i = 0; i < n - 1; i++)

{

int x = i;

for (j = i + 1; j < n; j++) //znajdz najmniejsza liczbe

{

if (tab[x] > tab[j])

x = j;

}

if (x != i) //jesli kolejna liczba nie jest najmniejsza to zamien z najmniejsza

{

int temp = tab[i];

tab[i] = tab[x];

tab[x] = temp;

}

}

}

int main()

{

int n=10;

int \*tab;

tab = (int\*) malloc (sizeof (int) \* n);

int i;

for(i=0;i<n;i++)

{

tab[i]=rand()%100;

}

SelectionSort(tab,n);

for(i=0;i<n;i++)

{

printf("%d\n",tab[i]);

}

free(tab);

return 0;

}

Zadanie nr 29.

Sortowanie szybkie

int Partition(int Tab[], int p, int r)

{

int bla;

int x = Tab[(p+r)/2];

int i = p - 1;

int j = r + 1;

int temp = 0;

while(1)

{

do

{

j = j - 1;

}

while(Tab[j] > x);

do

{

i = i + 1;

}

while(Tab[i] < x);

if (i < j)

{

temp = Tab[i];

Tab[i] = Tab[j];

Tab[j] = temp;

}

else

return j;

}

}

void QuickSort (int Tab[], int p, int r)

{

int q = 0;

if(p < r)

{

q = Partition(Tab, p, r);

QuickSort(Tab, p, q);

QuickSort(Tab, q+1, r);

}

}

int main()

{

srand(time(NULL));

int n=100000000;

int \*tab;

tab = (int\*) malloc (sizeof (int) \* n);

int i;

for(i=0;i<n;i++)

{

tab[i]=rand()%100;

}

QuickSort(tab,0,n-1);

for(i=0;i<n;i++)

{

printf("%d\n",tab[i]);

}

free(tab);

return 0;

}