/\*

include <stdio.h>

#include <stdlib.h>

#include <limits.h>

///1.Feladat, hány darab "i" és "y" van a szövegben ?

int f1(char \*p);

int main()

{

char s1[100] = "A koronavius megváltoztatta az egyutteles szabalyait";

int f1Eredmeny = 0;

f1Eredmeny = f1(s1);

printf("1. feladat: %d", f1Eredmeny);

return 0;

}

int f1 (char \*p)

{

int db = 0;

while(\*p)

{

if((\*p == 'i')|| (\*p == 'y'))

{

db++;

}

p++;

}

return db;

}

\*/

/\*

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

///2.Feladat: Tömbben a második, pozitív, négy decimális jegyû páratlan érték keresése

int fgv2(int \*p\_f2, int n2\_f2);

int main()

{

int t2[]={19905, -1060, 16807, 7651, 5096, -2404, -9668, 8991, -4467, 5922, +2347, -1195, -834, 6914, -2377, 2587, 7798, 9134, 6999, -3204};

int n2=sizeof(t2)/sizeof(t2[0]);

printf("\n A 2. feladat megoldasa: %d \n", fgv2(t2, n2));

return 0;

}

int fgv2(int \*p\_f2,int n2\_f2)

{

int szam\_f2=0, db\_f2=0;

while(n2\_f2)

{

if( ((\*p\_f2)>=1000) && ((\*p\_f2)<=10000) && ((\*p\_f2)%2!=0))

{

szam\_f2=\*p\_f2;

db\_f2++;

if (db\_f2==2)

break;

}

p\_f2++;

n2\_f2--;

}

return szam\_f2;

}

\*/

/\*

///3.Feladat: Tömbben a legnagyobb abszolút értékű elem megkeresése.

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

int fgv3(int \*p\_f3, int n3\_f3);

int main()

{

int t3[]= {-906,-1060,-6808,7651,5096,-2404,-9668,8991,-4467,5922,

-2347,-1195,-834,6914,-2377,2587,7798,9134,6999,-3204

};

int n3=sizeof(t3)/sizeof(t3[0]);

printf("\nA fgv3 megoldasa: %d\n", fgv3(t3, n3));

}

int fgv3( int \*p\_f3, int n3\_f3)

{

int max\_f3=INT\_MIN, abs=0, elojel=0, mentes=0;

while(n3\_f3)

{

//abszolut ertek kepzese

if(\*p\_f3<0)

{

abs=(\*p\_f3)\*(-1);

elojel=-1; //elojel

}

else

{

abs=\*p\_f3;

elojel=1;

}

if(abs>=max\_f3)

{

max\_f3=abs;

mentes=elojel; //elojel mentese

}

p\_f3++;

n3\_f3--;

}

max\_f3=max\_f3\*mentes;

return max\_f3;

}

\*/

/\*

///4.Feladat: Megkeresi a "c" betűket a tömbből és törli őket.

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

int fgv4( char \*p\_f4);

int main()

{

char s4[]="Ez nem volt egy nagyon bonyolult feladat, aminek a programja alig tizennyolc programsor.";

printf("\nA fgv4 megoldasa: %d\n", fgv4(s4));

printf("\nA fgv4 altal modositott szoveg: \n%s\n", s4);

return 0;

}

int fgv4( char \*p\_f4)

{

int db\_f4=0;

char \*j=0;

while(\*p\_f4)

{

if((\*p\_f4)=='c') //'c' betuk torlese

{

db\_f4++;

//\*p\_f4=' ';

j=p\_f4;

while(\*j) //sorfolytonossag biztositasa

{

\*j=\*(j+1);

j++;

}

}

p\_f4++;

}

return db\_f4;

}

\*/

///+1 Feladat sor:

//1.Ugyan az

/\*

///2.Feladat: Tömbben a harmadik, negatív, négy jegyű decimális jegyű páros szám megkeresése.

#include <stdio.h>

#include <stdlib.h>

int f2(int \*p, int n);

int main()

{

int t2[] = {-15248, -17912, -10564, 6487 ,2031 ,6884, 9493, -7451, -8196, 1597,-3350, -2056, -8806,-6619, 3304, -2984, 4266, 510, 5433, 3112, 549, -2202, 8355};

int f2Eredmeny = 0;

int n2 = sizeof(t2) / sizeof(t2[0]);

f2Eredmeny = f2(t2, n2);

printf("\n2. feladat: %d", f2Eredmeny);

return 0;

}

int f2(int \*p, int n)

{

int eredmeny = 0;

int i = 0;

int tmp = 0;

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

{

if((\*p <= -1000) && (\*p >= -9999) && ((\*p % 2) == 0))

{

tmp++;

}

if(tmp == 3)

{

eredmeny = \*p;

break;

}

}

return eredmeny;

}

\*/

/\*

///3.Feladat: Két legkisebb, páros értékű elem megkeresése.

#include <stdio.h>

#include <stdlib.h>

int f3(int \*p, int n);

int main()

{

int t3[] = {-906,-1060,-6808,7651,-10, 5096,-2404,-9668,12,8991,-4467,5922,-2347,-1195,-834, 6914,-2377,2587,-7798,9134,606, 6999,-3204};

int f3Eredmeny = 0;

int n3 = sizeof(t3) / sizeof(t3[0]);

f3Eredmeny = f3(t3, n3);

printf("\n3. feladat: %d", f3Eredmeny);

return 0;

}

int f3(int \*p, int n)

{

int eredmeny = 0;

int i = 0;

int \*tmp = 0;

int min = 0;

int min2 = 0;

tmp = p;

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

{

if((\*p % 2) == 0)

{

if(\*p < min)

{

min = \*p;

}

}

}

p = tmp;

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

{

if((\*p % 2) == 0)

{

if((\*p < min2) && (\*p > min))

{

min2 = \*p;

}

}

}

eredmeny = min + min2;

return eredmeny;

}

\*/

/\*

///4.Feladat: Ciklus, ami megkeresi és törli az "m" betűket a tömbből.

#include <stdio.h>

#include <stdlib.h>

int f4(char \*p);

int main()

{

char s4[100] = "Ez nem egy nagyon bonyolult feladat.";

int f4Eredmeny = 0;

f4Eredmeny = f4(s4);

printf("\n4. feladat: %d", f4Eredmeny);

return 0;

}

int f4(char \*p)

{

int db = 0;

int \*tmp = 0;

while(\*p)

{

if(\*p == 'm')

{

db++;

tmp = p;

while(\*tmp)

{

\*tmp = \*(tmp + 1);

tmp++;

}

}

p++;

}

return db;

}

\*/

/\*

///5.Feladat: Megkeresi a rendezett tömbben a beillesztendő új adat helyét és oda beszúrja azt úgy, hogy továbbra is rendezett maradjon.s

#include <stdio.h>

#include <stdlib.h>

int f5(int \*p, int n, int beszurando);

int main()

{

int t5[] = {113, 142, 289, 326, 337, 474, 555, 668, 730, 801};

int f5Eredmeny = 0;

int n5 = sizeof(t5) / sizeof(t5[0]);

f5Eredmeny = f5(t5, n5, 138);

printf("\n5. feladat: %d", f5Eredmeny);

return 0;

}

int f5(int \*p, int n, int beszurando)

{

int eredmeny = 0;

int i = 0;

int kieso = 0;

int beszur = beszurando;

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

{

if(beszur < \*p)

{

kieso = \*p;

\*p = beszur;

beszur = kieso;

}

}

eredmeny = beszur;

return eredmeny;

}

\*/