

ISTOTNE UWAGI DO PONIŻSZYCH ZADAŃ:

- Nie należy stosować zmiennych globalnych;
- Kod winien być pisany przejrzysto z uwzględnieniem wcięć podkreślających poziom polecenia;
- Od zadania nr 13 należy wprowadzić własne pliki nagłówkowe, zgodnie z treścią konkretnego zadania. Przy czym plików zawierających same funkcje NIE dołączamy za pomocą dyrektywy `include`, a kompilujemy osobno (na przykład: `gcc -c days.c`).
- Komplikacja programów testujących winna wyglądać tak jak poniżej:
`gcc -o zad13 zad13.c days.o`

Zad. 11.

Napisz funkcję wyświetlającą liczbę jej wywołań, a następnie krótki program sprawdzający poprawność jej działania (`zad11.c`).

```
void funkcja();
```

(Deklarację funkcji przedstawiono po prawej)

Zad. 12.

Napisz program (`zad12.c`) wyświetlający wszystkie możliwe złożenia podanej przez użytkownika liczby – kwoty w PLN w oparciu o tylko i wyłącznie banknoty **20, 50 i 100** PLN począwszy od banknotów o wyższym nominale, tak jak po prawej:

```
120 PLN = 1*100 + 0*50 + 1*20
120 PLN = 0*100 + 2*50 + 1*20
120 PLN = 0*100 + 0*50 + 6*20
```

Dodatkowe uwagi:

- Należy zaproponować optymalny algorytm nie stosując tablic;
- Program winien wypisać wszystkie możliwe złożenia zadanej kwoty;
- Należy użyć w zadaniu **nie więcej niż dwie pętle**.

Zad. 13.

Należy zdefiniować strukturę **DAY** zawierającą informację o dacie, a następnie dwuargumentową funkcję `days` zwracającą liczbę dni pomiędzy dwoma przekazanymi datami.

Stosowne deklaracje przedstawiono poniżej:

```
struct DAY;
int days( struct DAY, struct DAY );
```

Dodatkowe uwagi:

- Przed wykonaniem zadania należy zapoznać się parametrem `c` kompilatora `gcc`.
- Stosowne deklaracje oraz własną definicję struktury **DAY** należy umieścić w pliku nagłówkowym: **days.h**, zaś definicję funkcji `days` w osobnym pliku (`days.c`), a następnie napisać krótki program wykorzystujący utworzoną funkcję: `zad13.c`
- Proszę nie używać `typedef`
- Użytkownik może wprowadzać daty w dowolnej kolejności a funkcja w obu przypadkach musi zwracać poprawną wartość.
- Przy obliczaniu liczby dni należy uwzględnić lata przestępne oraz zmianę kalendarza ¹.
- Przykładowe wartości zwracane przez funkcję `days` dla danych dat:

2 lutego 2020	2 lutego 2020	0
12 grudnia 1234	20 stycznia 1410	63958
1 stycznia 1582	31 grudnia 1582	354

¹ Użyteczne linki:

https://pl.wikipedia.org/wiki/Daty_nowego_i_starego_porz%C4%85dku

<https://muzhp.pl/pl/e/51/wprowadzenie-w-polsce-kalendarza-gregoriańskiego>

Zagadnienia do przygotowania na najbliższe zajęcia:

- wskaźniki;
- przekazywanie argumentów do funkcji (wartość, wskaźniki i referencje);
- tablice statyczne
(jedno i wielowymiarowe; tworzenie, inicjowanie, przekazywanie do funkcji, etc.);
- tablica znaków a tekst;
- dynamiczna alokacja pamięci
(język C a C++; rezerwacja i zwalnianie, etc.);
- tablice dynamiczne;