① riadiace konštrukcie

→ transformácia na podmienené a nepodmienené skoky
→ väčšinu zvládate (dúfam), zaujímavé sú switch-case a zložené podmienky
vid. http://www.fit.vutbr.cz/~ibudisky/isu/2017/isu7-cv9.pdf

② isu8-cv9-1.asm LEN PRE GNU/LINUX !

→ prehľad: je nutné prilinkovať 32-bit knižnicu ncursesw

a) vlastný Linux: → nainštalujte balík libncursesw5-dev:i386 (Ubuntu) / lib32-ncurses (Arch)
→ pridajte -lncursesw ku kroku linkovania
(SASM: Settings → Build → Linking options)

b) školský CentOS 7: → stiahnite a rozbaľte statické knižnice, umiestnite ich k zdroj. súboru „isu8-cv9-1.asm"
http://www.fit.vutbr.cz/~ibudisky/isu/2018/centos74-ncursesw32.tar.gz
→ SASM: Settings → Build → Linking options pridajte
-L$SRCDIR$ -lncursesw -ltinfo -ldl

→ všeobecné pokyny:
→ STAV HRY je uložený v globálnej pamäti a registroch ESI, EDI, BX, CX. Ak tieto registre použijete pre iný účel, zálohujte ich.
- CX – smer pohybu hada
- BX – pozícia hlavy hada, BL = riadok, BH = stĺpec
- ESI – aktuálne skóre
- EDI – počet voľných polí v hre

→ REGISTER EAX nezálohuje žiadna funkcia (a ani sa to neočakáva)

→ Úloha 1

```
void GetInput() {                   // vystup cx
    do {
        GetChar();                  // vystup ax
        if (ax == CURSES_ERR)
            return;
    } while(ax < KEY_DOWN || ax > KEY_RIGHT);
    cx = ax;
}
```

→ Úloha 2

```
void UpdateHeadPosition() {     // vstup cx + bx, vystup bx
    switch(cx) {
    case KEY_UP:
        bl--;
        if (bl == 0)
            bl = HEIGHT - 2;
        break;
    case KEY_DOWN:
        bl++;
        if (bl == HEIGHT - 1)
            bl = 1;
        break;
    case KEY_LEFT:
        bh--;
        if (bh == 0)
            bh = WIDTH - 2;
        break;
    case KEY_RIGHT:
        bh++;
        if (bh == WIDTH - 1)
            bh = 1;
        break;
    }
}
```

→ Úloha 3

```
void GetTile() {                // vstup bx, vystup al
    al = playground[bl * WIDTH + bh];
}
```

→ Úloha 4

```
for(;;) {
    GetInput();                 // vystup cx
    UpdateHeadPosition();       // vstup cx + bx, vystup bx
    GetTile();                  // vstup bx, vystup al
    if (al == snek_food_int) {
        // aktualizace skore
        esi++;
        UpdateScore();
        // generovani noveho jidla
        PutFood();
    } else {
        // nejprve odstranime ocas
        RemoveTail();
        // musime znovu nacist interni reprezentaci
        GetTile();              // vstup bx, vystup al
        // test na konec hry
        if (al != ' ')
            break;
    }
    PlaceHead();                // vstup bx
    DrawAndSleep();
}
```

③ isa8-cv9-2.asm

→ dodržte volacin konvencin uvedenú v zadaní

→ výzva: skúste implementovať switch-case tabuľkou funkcií

```
int FmtPrint(const char* fmt, ...) {
    va_list ap;
    va_start(ap, fmt);

    while (*fmt) {
        if (*fmt == '%') {
            fmt++;
            switch (*fmt) {
            case '%':
                al = '%';
                WriteChar();
                break;
            case 'c':
                al = va_arg(ap, char);
                WriteChar();
                break;
            case 's':
                esi = va_arg(ap, const char*);
                WriteString();
                break;
            case 'u':
                eax = va_arg(ap, uint32_t);
                WriteUInt32();
                break;
            case 'd':
                eax = va_arg(ap, int32_t);
                WriteInt32();
                break;
            default:
                return -1;
            }
        } else {
            al = *fmt;
            WriteChar();
        }
        fmt++;
    }

    va_end(ap);
    return 0;
}
```