MUNI
FI

Cvičenie 07

Užívateľské dátové typy, dynamické štruktúry

enum

- Typ premennej s vopred danými pomenovanými hodnotami
- Stav procesu, deň v týždni, druh produktu
- Deklarácia – `enum nazov_typu { hodn1, hodn2, hodn3 };`
- Použitie – `enum nazov_typu nazov_premennej = hodn_enumu;`
- Interne reprezentované pomocou `int` – nezneužívať □□

enum

```c
enum status_t {
  pending,
  in_progress,
  finished
};

enum status_t update_process(int *progress, int curr_progress) {
  *progress += curr_progress;

  if (*progress <= 0)
    return pending;
  if (*progress >= 100)
    return finished;
  return in_progress
}
```

struct

- Typ premennej obsahujúci viacero zložiek
- Proces (stav, progress, názov), postava (meno, životy, sila útoku)
- Deklarácia – `struct nazov_typu {`

```
    typ1 nazov1;
    typ2 nazov2;
};
```

- Použitie – `enum nazov_typu nazov_prem = { hodn1, hodn2 };`
- Veľkosť – súčet veľkostí zložiek s prípadným zarovnaním (`sizeof`)

struct

```c
enum status_t {
  pending,
  in_progress,
  finished
};

struct process_t {
  char *name;
  int pid;
  enum status_t status;
  struct process_t *parent;
};

int main(void) {
  struct process_t p1 = {"bash", 42, in_progress, NULL};

  struct process_t p2;
  p2.name = "echo Denis <3";
  p2.pid = 1337;
  p2.status = pending;
  p2.parent = &p1;

  // Ostatné zložky neinicializované
  struct process_t p3 = {.pid=420, .status=finished};
}
```


MUNI FI

`struct`

- Hodnotové typy:
  - Použitie ukazovateľov pri predávaní medzi funkciami
  - Pri kopírovaní structov obsahujúcich ukazovatele sa skopírujú len adresy
- Prístup k položkám:
  - `.` – Hodnota typu `struct`
  - `->` - Hodnota typu ukazovateľ na `struct` (`(*x).attr == x -> attr`)
- Možné normálne dynamicky alokovať

struct

```c
enum status_t {
  pending,
  in progress,
  finished
};

struct process_t {
  char *name;
  int pid;
  enum status_t status;
  struct process_t *parent;
};

void print_callers(struct process_t *process) {
  struct process_t *current = process;
  while (current != NULL) {
    printf("(%d) %s", current->pid, current->name);
    current = current->parent;
    if (current != NULL) {
      printf(" <- ");
    }
  }
}

int main(void) {
  struct process_t p1 = {"bash", 42, in progress, NULL};
  struct process_t p2 = {"./a.out", 1337, in_progress, &p1};
  struct process_t p3 = {"echo Denis <3", 85, finished, &p2};

  print_callers(&p3);
  // (85) echo Denis <3 <- (1337) ./a.out <- (42) bash
}
```


MUNI FI

`typedef`

- „Typový alias“
- Používané pre skrátenie typu alebo abstrakciu od typu
- Zavedenie: `typedef stary_typ novy_typ`

typedef

```c
enum status_t {
  pending,
  in_progress,
  finished
};


struct process_t {
  char *name;
  int pid;
  enum status_t status;
  struct process_t *parent;
};

enum status_t CURRENT_STATUS = pending;
struct process_t CURRENT_PROCESS = {
  "bash",
  42,
  in_progress,
  NULL
};
```

```c
enum status_t {
  pending,
  in_progress,
  finished
};
typedef enum status_t status_t;

typedef struct process_t {
  char *name;
  int pid;
  enum status_t status;
  struct process_t *parent;
} process;

status_t CURRENT_STATUS = pending;
process CURRENT_PROCESS = {
  "bash",
  42,
  in_progress,
  NULL
};
```


MUNI FI

Jako?
???