**BI-SPOL-18 Datové typy v programovacích jazycích. Staticky a dynamicky alokované proměnné, spojové seznamy. Modulární programování, procedury a funkce, vstupní a výstupní parametry. Překladač, linker, debugger**

BI-PA1

**Datové typy v programovacích jazycích**

Proměnná může uchovávat různé informace. Dle typu informace musíme alokovat paměť – určíme počet bytu (asociace s názvy – int, char, …). U slabě typovaných jazyků není potřeba uvádět typ, o přiřazení paměti se starají interní procesy

* proměnná může uchovávat různé informace
* podle typu je třeba také alokovat paměť
* u některých jazyků není třeba definovat typ, o přiřazení paměti se stará interní proces
* do paměti se ukládají dle little/big endian (liho/holi)
* primitivní typy: např char, (un)signed int, long, float, short, double, bool
* kontejnery: vektor, mapa, list…
* programátor si může deklarovat vlastní datové typy (class, struct...)

napsat interpretaci v bajtech v C v paměti

**Ukazatel**

* slouží k uložení adresy v paměti počítače (na dané adrese se nacházejí nějaká data)
* proměnná obsahující odkaz na datovou buňku v paměti

operace s pointery - ukazky

**Reference**

* odkaz na proměnnou nebo instanci objektu
* abstraktnější varianta ukazatele
* neobsahuje žádnou informaci o uložení objektu v paměti

**Staticky alokované proměnné**

* definuje předem daný počet bytů
* alokace na zásobníku
* vzniká deklarací

**Dynamicky alokované proměnné**

* dopředu nelze říci, kolik místa daná informace zabere (např. pole)
* místo se přiřazuje v době běhu programu
* na místo se přistupuje ukazatelem
* alokace na haldě
* programátor se musí o paměť starat (new, free) nebo čekat na garbage collector

**Spojové seznamy**

* dynamicky alokovaná struktura
* každý prvek obsahuje hodnotu a ukazatel na další prvek (pokud oboustranný, tak i na předchozí)
* cyklický nebo poslední prvek končí odkazem NULL
* přidání O(1), odebrání a find O(n) v případě neseřazeného

**Modulární programování**

* jeden ze způsobů strukturování programů
* program je rozdělen do několika souborů (modulů), ty se dají vyměnovat
* lepší udržovatelnost, přehlednost a logické oddělení
* k modulům se přistupuje jako k black boxu s veřejným interfacem
* každý modul má vlastní funkce, třidy, metody… jiné využití

**Funkce**

* podprogram, který na základě vstupních proměnných řeší konkrétní podproblém
* svůj výsledek předává ve formě výstupních parametrů
* při volání je vytvořen aktivační záznam na zásobníku

**Procedury**

* funkce bez návratové hodnoty (vrací *void*)

**Vstupní parametry**

* proměnné, které vstupují do konkrétní funkce
* mají svůj název (většinou i typ) a mohou mít defaultní hodnotu
* kopírují se nebo se předávají ukazatelem (kopíruje se adresa na buňku)

**Výstupní parametry**

* proměnné které vystupují z funkce

**Překladač (kompilátor)**

* nástroj pro překlad vstupního (vyššího) jazyka do jazyka nižšího
* nejčastějí překlad do strojového jazyka
* optimalizuje kód
* v případě existence reference na jiné soubory se předávají do linkeru

**Linker**

* řeší reference mezi jednotlivými moduly, soubory a knihovnami
* propojuje zkompilované soubory do funkčního celku
* výstup je spustitelný binární soubor

**Debugger**

* nástroj pro ladění kódu
* umožňuje třeba krokování v programu
* slouží k lepšímu porozumění chování programu

pojmy:

* adresa – jednoznačná identifikace bajtu v paměti
* procesor má typicky přístup k bajtům, nikoliv k jednotlivým bitům
* operační paměť – uchovává program a data v době běhu programu
* člověk pracuje s informacemi (celé čísla, znaky,…) – aby mohla být takováto informace uložena do paměti, musí být zakódována do jednoho či více bajtů

otazky

* zaměřte se na ukazatel, jeho vlastnosti a použitelnost
* little/big endian – vysvětlit
* globální/lokální proměnné
* co je to garbage collector
* co se stane když globální proměnnou jen deklarujeme ale neinicializujeme
  + globální proměnné jsou vždy inicializovány nulovými bajty