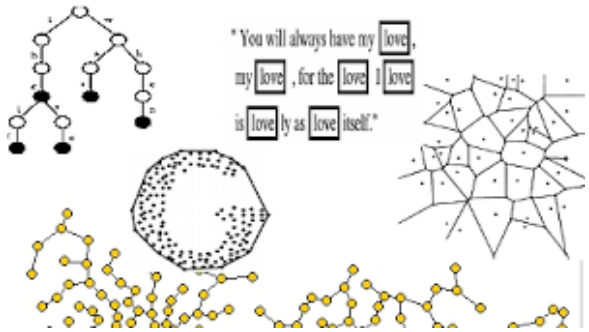
Estrutura de Dados

Ponteiro

Alexandre Ribeiro

Aula 5 – Alocação Dinâmica



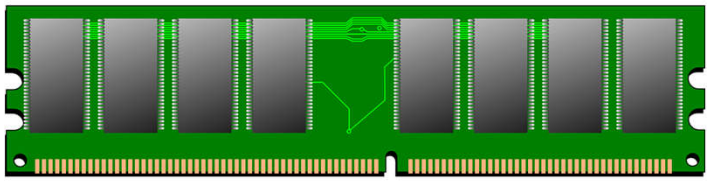
Alocação de memória do S.O

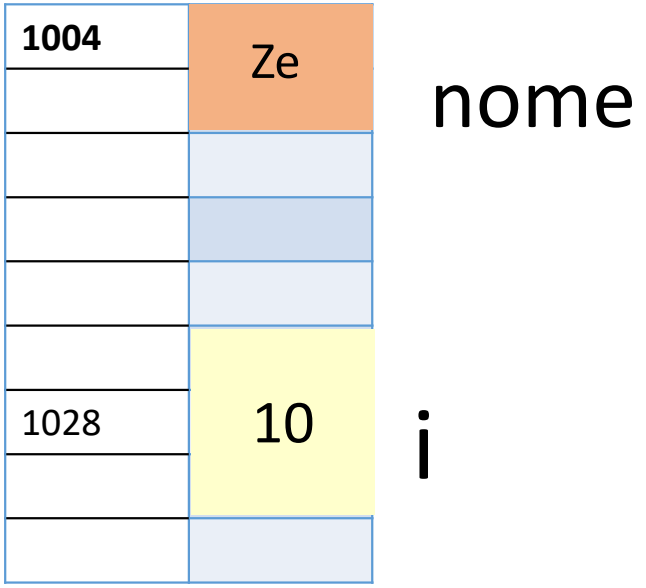
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5
6     int i;
7     char nome[20];
8     float numero [2];
9
```

Tipo	Tamanho (em bits)	Intervalo
char	8	-128 a 127
int	16	-32768 a 32767
float	32	3,4E-38 a 3,4E+38
double	64	1,7E-308 a 1,7E+308
void	0	sem valor

i = 10

nome = ze





Ponteiros

Um ponteiro é uma variável capaz de armazenar um endereço de memória de uma outra variável.

Ponteiros

Como declarar um ponteiro?

```
int    *p;          // ponteiro para inteiro

char  *x;           //ponteiro para char

float  *z;          //ponteiro para float

pessoa *p;     // ponteiro para uma struct pessoa
```

Alocação Dinâmica

Definição

- A *alocação dinâmica* permite ao programador criar "variáveis" em tempo de execução, ou seja, alocar memória para novas variáveis quando o programa está sendo executado, e não apenas quando se está escrevendo o programa.

  - Quantidade de memória é alocada sob demanda, ou seja, quando o programa precisa
  - Menos desperdício de memória
    - Espaço é reservado até liberação explícita;
    - Depois de liberado, estará disponibilizado para outros usos e não pode mais ser acessado;
    - Espaço alocado e não liberado explicitamente é automaticamente liberado ao final da execução

Alocação Dinâmica

- A linguagem C ANSI usa apenas 4 funções para o sistema de alocação dinâmica, disponíveis na stdlib.h:
  - malloc
  - calloc
  - realloc
  - free

Alocação Dinâmica - MALLOC

- **malloc**
  - A função malloc() serve para alocar memória e tem o seguinte protótipo:

```
void *malloc (unsigned int num);
```

- Funcionalidade
  - Dado o número de bytes que queremos alocar (**num**), ela aloca na memória e retorna um ponteiro **void*** para o primeiro byte alocado.

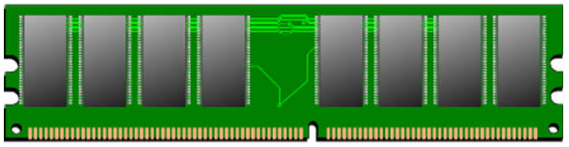
Alocação Dinâmica

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

    malloc(4);
    sizeof(int);
```

1004
1008
1012
1016
1020
1024
1028
1032
1036
1040
1044
1046
1050
1052
1056



Alocação Dinâmica

- Alocar 1000 bytes de memória livre para char.

```
char *p;
p = (char *) malloc(1000);
```

Alocação Dinâmica

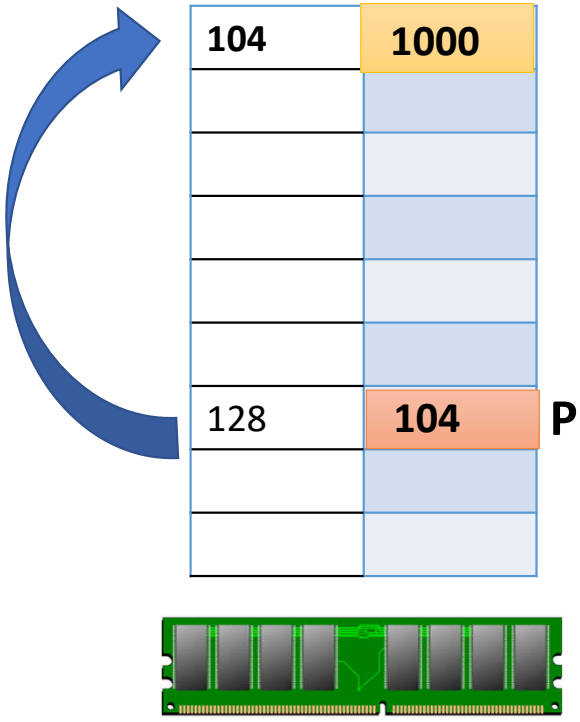
- Alocar tamanho de um inteiro de memória livre para inteiro.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int *p;
    p=(int*)malloc(sizeof(int));
    *p=1000;

    printf("%d",*p);

    free(p);
```



Alocação Dinâmica de Vetores

- Para armazenar um VETOR o compilador calcula o tamanho, em bytes, necessário e reserva posições sequenciais na memória;

- Existe uma ligação muito forte entre ponteiros e vetores.
  - O nome do array é apenas um ponteiro que aponta para o primeiro elemento do array.

Alocação Dinâmica de Vetores

- Alocar espaço para vetor de 50 posições do tipo inteiro:

```
int *p;
p = (int *) malloc(50*sizeof(int));
```

p=

1000	1002	1003	1004	1005	1006	1007	...	1050
5	6	7	8	9	10	11	12	19

p[ ]

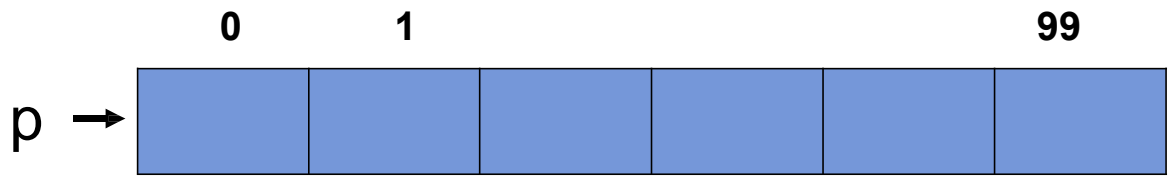
Alocação Dinâmica de Vetores

- Alocar espaço para vetor de 100 posições do tipo inteiro:

```
int *p;
int i, N = 100;

p = (int *) malloc(N*sizeof(int));

for (i = 0; i < N; i++)
    scanf("%d",&p[i]);
```



Alocação Dinâmica de Vetores

- Se não houver memória suficiente para alocar a memória requisitada, a função **malloc()** retorna um ponteiro nulo

```
int main(){
    int *p;
    p = (int *) malloc(5*sizeof(int));
    if(p == NULL){
        printf("Erro: Memoria Insuficiente!\n");
        system("pause");
        exit(1);
    }
    int i;
    for (i=0; i<5; i++){
        printf("Digite o valor da posicao %d: ",i);
        scanf("%d",&p[i]);
    }

    return 0;
}
```

Alocação Dinâmica – FUNÇÃO CALLOC

- **calloc**
  - A função calloc() também serve para alocar memória, mas possui um protótipo um pouco diferente:

```
void *calloc (unsigned int num, unsigned int size);
```

- Funcionalidade
  - Basicamente, a função calloc() faz o mesmo que a função malloc(). A diferença é que agora passamos a quantidade de posições a serem alocadas e o tamanho do tipo de dado alocado como parâmetros distintos da função.

Alocação Dinâmica – FUNÇÃO CALLOC

```c
int main(){
    //alocação com malloc
    int *p;
    p = (int *) malloc(50*sizeof(int));
    if(p == NULL){
        printf("Erro: Memoria Insuficiente!\n");
    }
    //alocação com calloc
    int *p1;
    p1 = (int *) calloc(50,sizeof(int));
    if(p1 == NULL){
        printf("Erro: Memoria Insuficiente!\n");
    }

    return 0;
}
```

Alocação Dinâmica – FREE

- **free**
  - Diferente das variáveis definidas durante a escrita do programa, as variáveis alocadas dinamicamente não são liberadas automaticamente pelo programa.
  - Quando alocamos memória dinamicamente é necessário que nós a liberemos quando ela não for mais necessária.
  - Para isto existe a função **free()** cujo protótipo é:

```
void free(void *p);
```

Alocação Dinâmica – Registro

- **Struct**
  - Aloca memória para um ponteiro do tipo da struct
  - Utiliza o operador seta (->) para acessar o conteúdo.
  - Utiliza se o free para liberar a memória alocada.

```
struct cadastro{
    char nome[50];
    int idade;
};

int main(){
    struct cadastro *cad = (struct cadastro*) malloc(sizeof(struct cadastro));
    strcpy(cad->nome,"Maria");
    cad->idade = 30;

    free(cad);

    return 0;
}
```

- 1) Construa um programa (main) que aloque em tempo de execução (dinamicamente) um vetor V1 de 3 posições (#define 3) usando a chamada da função malloc.
- Agora, aproveite este programa para construir uma função (recebe_vetor) que recebendo os parâmetros do vetor;

1) aloque um outro vetor de 3 posições dinamicamente na função;
2) multiplique as posições dos dois vetores;